EMBEDDED APPLICATIONS JOURNAL

THE EMBEDDED MICROCONTROLLERS AND PROCESSORS APPLICATIONS SUPPORT PUBLICATION

Q4, 1992

BACK ISSUES Q1 92 Q4 92

intel

Increased Performance and More Memory for 87C196 Applications

Intel recently introduced the 87C196KD as its latest addition to the 16-bit microcontroller family, and a 20-MHz in-circuit emulator (ICE™) support tool. The 87C196KD is a memory upgrade that retains C196 code and architecture compatibility and is available in both 16 and 20 MHz versions. The ICE provides enhanced functionality (including 20 MHz real-time emulation and on-circuit emulation) and supports the new 8xC196KD microcontroller.

The 87C196KD is a memory upgrade to Intel's existing 87C196KC with 32Kbytes on-chip ROM and one Kbyte on-chip RAM. The availability of more on-chip ROM helps to eliminate off-chip ROM, frees board space and reduces system cost. More on-chip RAM provides additional registers which allow faster access to data and eliminates 32K OTPROM. External EPROM in designs requiring less than 32 Kbytes code is no longer needed and programming in high-level languages is more practical. The 20MHz part gives an across-the-board 25 percent performance increase over the current KC product.

The ICE-196KD/HX features on-circuit emulation enabling the developer to debug circuits with surface-mounted components. There is no need to remove the component from the printed circuit board. The ICE-196KD/HX emulator does real-time emulation at full component speed, allowing applications to be debugged under realistic conditions.

The ICE-196KD/HX aids the designer by simplifying debugging, speeding time-to-market and aiding code development. Powerful break, trace and other debug features shorten design cycles by helping both hardware and software designers find and correct elusive bugs quickly.

Call 800-548-4725 for more information on the 87C196KD or the ICE-196KD/HX.

MORE COPIES

NEED MORE COPIES OF the first issue of the *Embedded Applications Journal*?
Call 800-548-4725 and order #241294-001.
Europe and other international locations, please contact your local sales office or distributor for additional copies.

WHAT'S INSIDE

ACE - New Name, New Look

Intel has added on-line reference capabilities under the Windows environment to create a more powerful embedded controller programming tool. ApBUILDER takes over where ACE left off, providing a learning vehicle, reference guide and programmer all-in-one. ApBUILDER is now available for Intel's 80C186EA/EB/EC 16-bit microprocessor products. The 80C196KB/KC/KD and 8XC194/98 version will be available in October, 1992. ACE is still available for the MCS®-51 family and the MCS®-96 family products not currently supported by ApBUILDER.

ApBUILDER has pull down menus to allow the user to view product features, data sheets, fact sheets and answers to commonly asked questions with a click of a

Continued on Page 2.

NEXT ISSUE:

- Micro Stepping Motor Control Using the 87C196KD
- Interfacing to the 82527 CAN Device
- PTS and the 87C196KD
- Floppy Drive Interface for the 80C186Ex

Continued From Page 1.

button. Icons are used to represent specific software functions. A key feature of ApBUILDER is the ability to generate Assembler code from three complexity levels: Instruction, Register, and Peripheral. Instruction coding provides timing information for the selected instruction and prompts the user with destination and source options. Register coding allows the user to select the configuration bit by bit. Peripheral coding establishes peripheral functions on a much higher level. Once determined, all of the necessary registers and instructions are automatically programmed to support the chosen configuration. ApBUILDER requires Windows 3.0, 386 based PC, VGA color monitor, a mouse, 4 meg of memory, 3.5" floppy and 16 meg of hard disk memory to run. The program loads on the user's system with a few simple commands. Helpful hints on how to use ApBUILDER are available on FaxBACK™ (800) 628-2283 or (916) 351-3105, Document number 2166.

"Hypertext" manuals (electronic versions of user's manuals) are also available. Hypertext manuals allow word searches and display summary lists, eliminating the need to look through pages and pages of reference literature to find a specific piece of information. Hypertext manuals may be accessed through ApBUILDER or through the Windows Help file.

Interested in receiving a copy? Call (800) 548-4725 or (408) 765-1459 to order. Order numbers are listed below.

ApBUILDER and hypertext for 80C186 272171
ApBUILDER for 186 plus C196KB/KC/KD.....272216

EDITOR: STEVEN M. MCINTYRE
ASSISTANT EDITOR: PHYLLIS STERN
PRODUCTION: DOUG CHILDS, CHRIS PARSONS

New Embedded Control FaxBACK™ Documents

As of June 26, 1992

Title	Document No.
ApBUILDER Information and Order Package	2508
ApBUILDER Rel 1.0 Anomalies - April 1992	2166
Three Volt Vendor Summary - April 1992	2152
MCS-51 CHMOS Programming Documents	2151
8051/31AH Shrink C-Step External Interrupt 0 Errata	2161
EV80C186EB Evaluation Board Errata	2158
Paradigm Software Details for 80186	2509
MBE196KR Multi-Board Emulator Fact Sheet	2162
8XC196KR Unsupported Programming Modes Notification	2160
8X9XBH, KB, KC & KD Serial Port Mode 0 - App Brief	2159
87C196KR Auto Programming (Version 2)	2149

NEW PHONE NUMBERS FOR FaxBACK APPLICATION SUPPORT

FaxBACK has moved. The new numbers are 800-628-2283 or 916-351-3105.

From the editor

The response to the first issue of the Embedded Applications Journal (EAJ) has been great. Thanks to all our readers for the positive and the negative feedback. Keep it coming! I need your suggestions to ensure the EAJ is meeting the needs of our readers. Please tell us what you liked, disliked, and articles you would like to see in the future. Simply fill out the questionnaire found on the back cover of the EAJ and mail or fax to the Assistant Editor's attention at (800) 722-2862 or (602) 554-7436. Every completed questionnaire receives a FREE gift.

The second issue features some exciting articles. Key articles include a new product announcement for the 87C196KD, introduction of the ApBUILDER programming tool, interfacing the 8XC196KR 16-Bit microcontroller to a personal computer, and designing with 3 volt/5 volt mixed voltage to name a few.

Please pass this publication on to interested colleagues. Again, I welcome feedback and suggestions to make this publication a valuable tool.

Steven M. McIntyre
Applications Manager
Embedded Microcomputer Division

Interfacing the 8XC196KR 16-Bit Microcontroller to a Personal Computer

Bob Johnson
Applications Engineer
Intel

Introduction

The subject of this application brief is interfacing the 8XC196KR to the ISA (Industry Standard Architecture) bus on a PC or AT computer. Throughout the rest of this application note, the 8XC196KR will be referred to as the KR. The Slaveport is a peripheral of the KR that enables it to sit on a bus and be treated as any other memory mapped device. The Slaveport is ideally suited for interfacing to a microprocessor such as the X86 family. Placing the KR on the ISA bus gives the PC applications developer access to a powerful 16-bit microcontroller. The interface is simple, yet gives direct access to all the peripherals on the 8XC196KR.

Throughout this application note, program examples are referred to in the appendix. Due to the length of the program examples, the appendix is not included in this document, but can be found on FaxBACK.

This application note begins with an overview of the Slaveport. More information on the Slaveport, or any other KR circuitry can be found in the KR Users Manual (Order #270952). Details explaining three of the four methods of communicating with the Slaveport follow the overview.

Many peripherals and features exist on the KR. These include:

1) Eight 10-bit A/D channels
2) Two Synchronous Serial Channels (SSIO)
3) A full duplex UART
4) Ten channel Event Processor Array
5) 37 interrupts
6) Idle and Powerdown modes to save power consumption
7) On-board programmable ROM
8) Executable (Data) RAM
9) Register (Data) RAM
10) Windowing functions - Windowing allows fast and efficient access to either SFRs or additional RAM.

Overview

Enabling the Slaveport forces RD# (SLPRD#), WR# (SLPWR#), and ALE to take on new functions as inputs. In almost all other microcontrollers, these pins are outputs only. Additionally, a CS# (SLPCS#) input and an interrupt output (SLPINT) complete the bus interface. Figure 1 shows a block diagram of the internal Slaveport.

The Slaveport is capable of interfacing to both multiplexed and non-multiplexed address/data buses. The Slaveport uses only one address bit called SLP_ADDR. The source for SLP_ADDR is either ALE or P3.1. SLPL, a bit in the Slaveport Control register (SLP_CON), determines which pin is the source for SLP_ADDR. If SLPL = 0, ALE is the source of SLP_ADDR. Use this configuration for a non-multiplexed bus. If SLPL = 1, P3.1 is the source for SLP_ADDR and ALE latches P3.1. Use this configuration for a multiplexed bus. Table 1 shows how the Slaveport combines SLP_ADDR with SLPRD# or SLPWR# to determine the register accessed by the master device. In all cases, Port 3 is a bi-directional data bus through which each data transfer takes place.

Table 1. Slaveport Register Decoding

SLP_ADDR	RD#	WR#	Active Register
0	0	1	P3_REG
0	1	0	P3_PIN
1	0	1	SLP_STAT
1	1	0	SLP_CMD

Hardware Interface

Minimal hardware is necessary to interface to the ISA bus. Once the Slaveport has been enabled, the only additional component required is for address decoding. The interface used in this application note is shown in Figure 2. The KR Port 3 is connected directly to the lower 8 bits of the ISA data bus. All data transfers are through port 3. Address lines A1-A9 and AEN# are connected to the PAL to generate chip-select for the Slaveport. The output of the PAL is connected directly to the SLPCS# input on the KR. SLPWR# and SLPRD# are qualified with chip-select and connected directly to the ISA WR# and RD# signals. The qualification is to eliminate invalid responses to these signals by the KR. ALE is connected directly to bit 0 of the ISA address bus. This signal determines which register pairs (SLP_CMD/P3_REG or SLP_STAT/P3_PIN) are accessed. P2.2 on the KR is tied to Reset on the ISA bus. Reset signifies when the power-on sequence ends. P2.2 is used only if the KR is powered externally from the ISA bus and supplies its own reset.

The component used for address decoding is a 85C22V10, PLD. The addresses chosen for the KR are 210H and 211H. The PLD decodes addresses 210H and 211H to generate the chip-select for the Slaveport. The entire program used for the 85C22V10 is shown in appendix A. The main rea-

son a 85C22V10 was chosen is the requirement for a 10 product sum.

Program Examples

These examples show techniques that allow one to experiment with the four basic interface registers (P3_PIN, P3_REG, SLP_STAT, and SLP_CMD). Reading and writing the four registers is the first step in making a complete interface. Using the following methods as a starting point enables a person to create solutions to various interface requirements. In some systems, tying SLPINT to one of the available PC interrupt inputs would allow an interrupt driven system. The examples in this application note all use a polling technique.

Developing the code for these three examples requires two key steps. The first step develops the MCS®-96 code used for the KR and the second step creates the x86 code used for the PC ISA bus. In all the examples, after assembling the x86 code it is linked to create an executable file. Running the executable file on the PC controls the communication to the KR sitting on the ISA bus.

Example 1A - Reading the KR Input Buffer

First, initialization of Port 2 and Port 5 takes place. P2.2 monitors the Reset signal on the ISA bus to determine when the x86 processor is up and running. This avoids unwanted control signals that might occur as the PC powers up. A Reset is considered valid when it is high for 64mS and then remains low for at least 128ms (at 16MHz KR oscillator frequency). Port 5 is initialized so all Slaveport control pins are inputs and SLPINT (P5.4) is an output. This also disables all the alternate functions on Port 5.

Initialization of SLP_CON occurs after a valid Reset. Figure 3 shows SLP_CON. The KR enables the Slaveport, determines how ALE functions, and which interrupts affect SLPINT. The address bit comes through ALE so bit 2 is

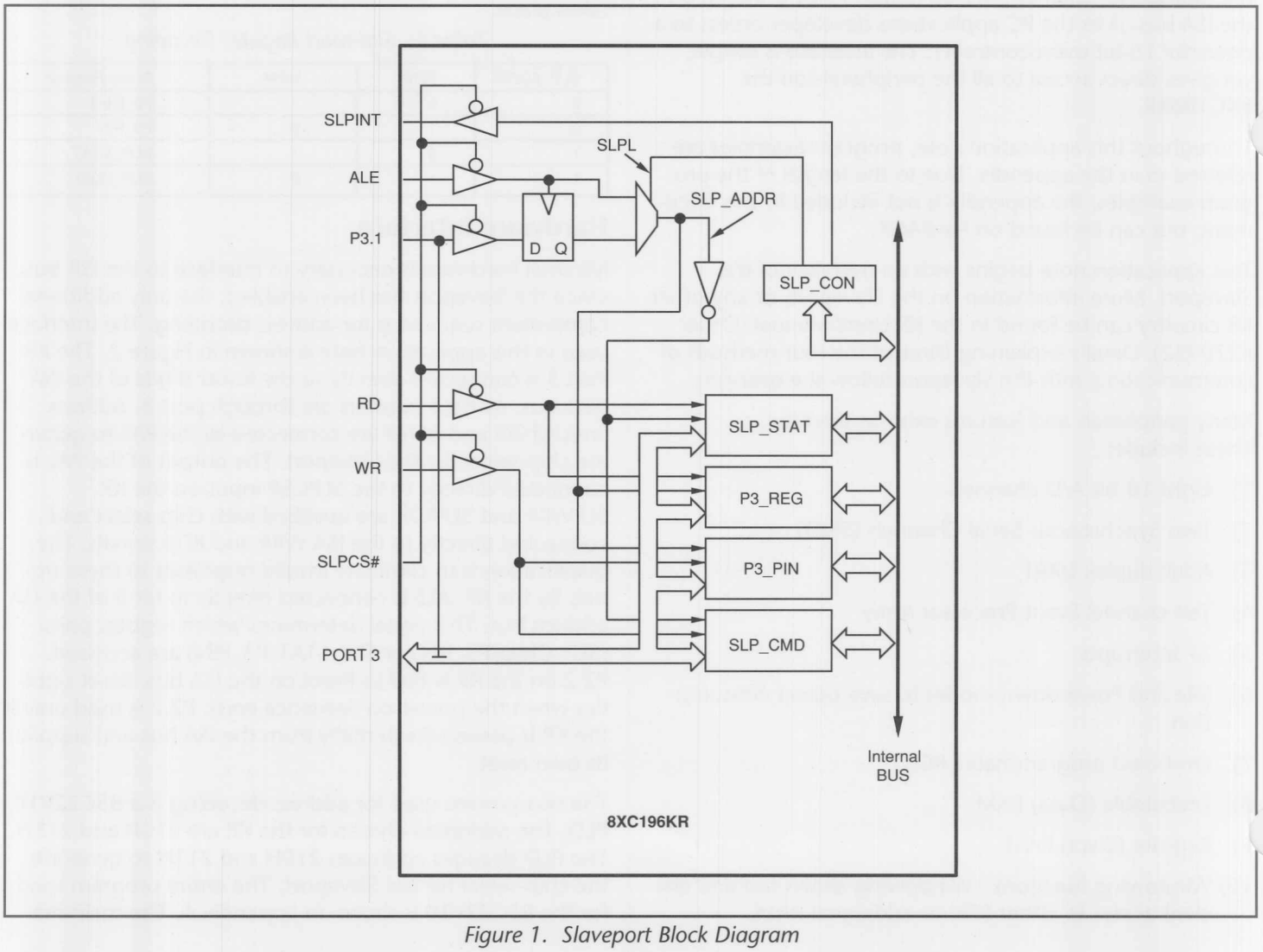


Figure 1. Slaveport Block Diagram

cleared. The flags IBE (Input Buffer Empty) and OBF (Output Buffer Full) can force SLPINT high, although this pin is not used. Note that these flags have opposite meaning from the interrupts IBF (Input Buffer Full) and OBE (Output Buffer Empty). To insure against the possible corruption of the Slaveport command and status SFRs, these registers are read or loaded with 0.

SLP_CON

0	0	0	0	SLP	SLPL	IBEmask	OBFmask

SLP -	Slaveport Enable (1 = enabled)
SLPL -	Slaveport Latch (1 = ALE is latch signal for address(multiplexed bus), 0 = ALE is address input)
IBEmask -	Input Buffer Empty (1 = SLPINT is high when input buffer is empty)
OBFmask -	Output Buffer Full (1 = SLPINT is high when output buffer is full)

Figure 3. SLP_CON

Finally, the IBF interrupt is enabled. This is the only interrupt used. Writing to the Slaveport input buffer (P3_PIN) sets the IBF interrupt pending flag. The program executes a tight loop until an interrupt occurs. For the interrupts to interface with the demo board monitor, the interrupt vectors associated with the Slaveport were changed to:

OBE interrupt -> 200CH
IBF interrupt -> 200EH
CBF interrupt -> 2030H

When an interrupt occurs:

1) The interrupts are disabled
2) The data is read
3) A DJNZ loop is executed. (This is placed here only to show some sample code running. In a real application this code would be replaced with a routine that does something useful with the data that has just been read.)

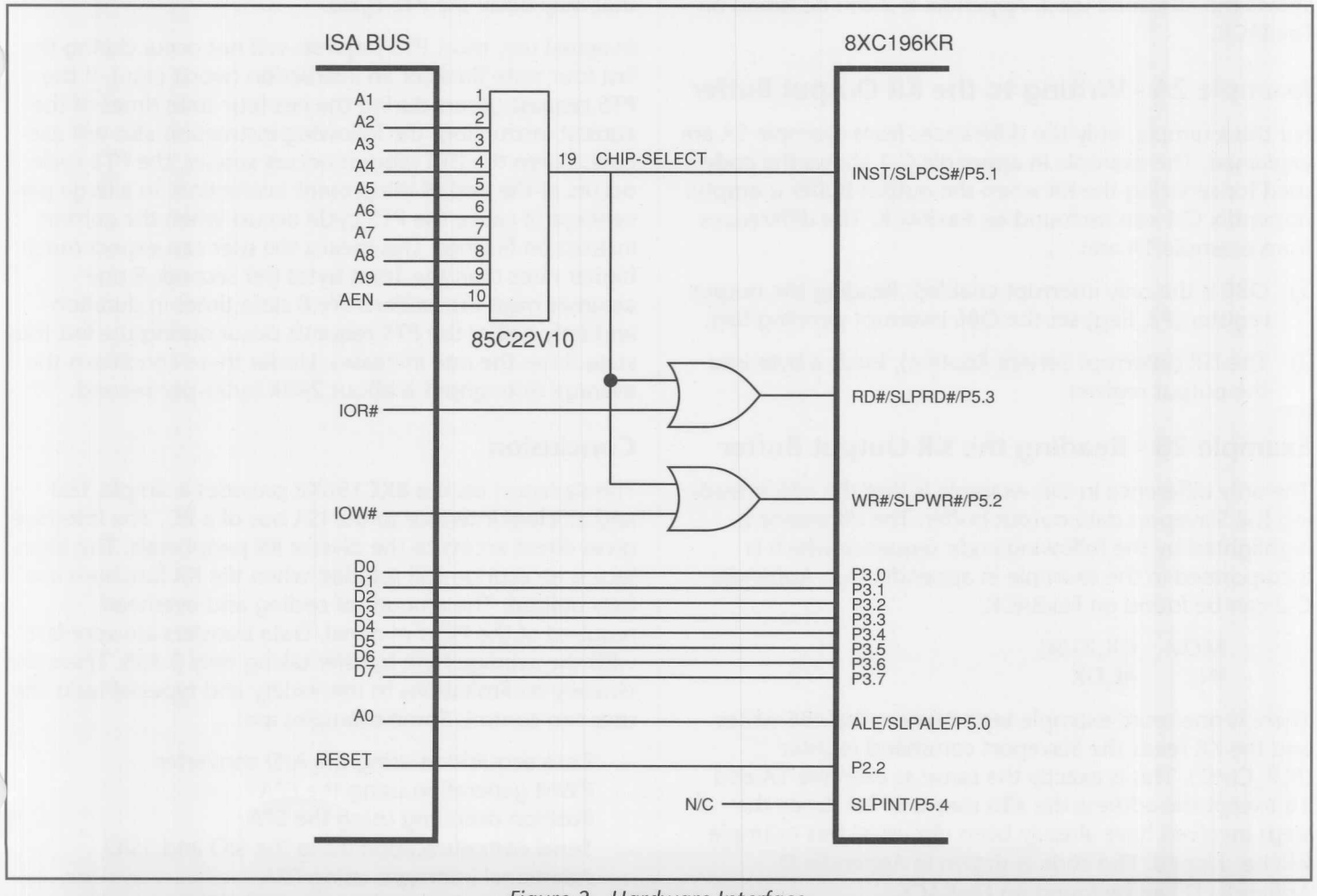


Figure 2. Hardware Interface

4) The interrupts are enabled

5) The program returns to the Main_Loop.

The program in appendix B-1 contains the KR code used to read the Slaveport input buffer. Appendix B-1 can be found on FaxBACK.

Example 1B - Writing the KR Input Buffer

The x86 communicates to the KR using "IN" and "OUT" instructions with 210H or 211H as the port address. 210H is the address for the P3_REG/P3_PIN registers and 211H is the address for the SLP_STAT/SLP_CMD registers. In this example the x86 is writing to the P3_PIN register so the code sequence is:

```
MOV   AL,XXH
MOV   DX,210H
OUT   DX,AL
```

Also included in the x86 code is a sequence to see if a key is pressed on the keyboard. This is used to exit the x86 program. Other than this, the x86 program only loops on the output instructions but changes the data for every-other output. The program example in appendix B-2 shows the x86 code used. Appendix B-2 can be found on FaxBACK.

Example 2A - Writing to the KR Output Buffer

For this example, only the differences from example 1A are explained. The example in appendix C-1 shows the code used for servicing the KR when the output buffer is empty. Appendix C-1 can be found on FaxBACK. The differences from example 1A are:

1) OBE is the only interrupt enabled. Reading the output register (P3_Reg) set the OBE interrupt pending flag.

2) The ISR (Interrupt Service Routine), loads a byte into the output register.

Example 2B - Reading the KR Output Buffer

The only difference in this example is that the x86 is reading the Slaveport data output buffer. The difference is highlighted by the following code sequence which is incorporated in the example in appendix C-2. Appendix C-2 can be found on FaxBACK.

```
MOV   DX,210H
IN    AL,DX
```

There is one more example tested where the x86 writes and the KR reads the Slaveport command register (SLP_CMD). This is exactly the same as example 1A and 1B except the address the x86 uses is 211H. Since the steps involved have already been discussed this example will be skipped. The code is shown in Appendix D. Appendix D can be found on FaxBACK.

Maximum Data Transfer Rates

To gain the most efficient and fastest transfer rates requires the use of the PTS. The examples used so far in the application note do not make use of the PTS but do use the interrupts associated with the Slaveport. This section assumes the PTS is used. The fastest, worst-case, theoretical transfer rate for back-to-back exchanges is about 131K bytes-per-second. By not using the NORML instruction, the rate increases to about 166K bytes-per-second. The PTS response and overhead time limits the transfer rate.

The worst case access time equals:

((4 state times) + an instruction execution (30 -39 state times)) + PTS transfer time (18 state times).

These numbers assume:

A) The crystal frequency is 16MHz

B) The PTS handles all data transfers

C) No other activities will delay the PTS cycle

D) The PTS bug is fixed (as it is on C-step and later device).

The 8XC196KR Users Manual lists the possible conditions that may delay the PTS cycle.

In actual use, most PTS requests will not occur during the last four state times of an instruction (worst case). If the PTS request occurs during the last four state times of the current instruction, the following instruction also will execute. When the PTS request occurs sooner, the PTS cycle occurs at the end of the present instruction. In a large percentage of cases, the PTS cycle occurs when the current instruction finishes. This means the user can expect much higher rates than the 166K bytes per second. If one assumes most instructions are 8 state times in duration and only half of the PTS requests occur during the last four state times the rate increases. Under these conditions the average throughput is about 294K bytes-per-second.

Conclusion

The slaveport on the 8XC196KR provides a simple, fast and efficient interface to the ISA bus of a PC. The interface gives direct access to the diverse KR peripherals. The interface is an economical solution when the KR functions are fully utilized. The amount of coding and overhead required of the PC is minimal. Data transfers are very fast with the average byte transfer taking only 3.4µS. There are virtually no limitations to the variety and types of tasks the user can control. Some examples are:

Data acquisition using the A/D converter
PWM generation using the EPA
Position decoding using the EPA
Serial communication using the SIO and SSIO
Additional interrupts using EPA
Parallel interfaces using standard I/O port

General application control using standard I/O ports (such as EPROM Programmers)

The only piece missing is the actual handshake between the ISA bus and the KRs Slaveport.

Top Five Issues

1) All Port 5 pins (except SLPINT) must be configured as inputs with their alternate functions disabled immediately after the PC comes out of reset. Bus conflicts occur if RD# or WR# are outputs and the PC will not boot up.
2) Disable the Slaveport until after the PC is up and running. Erratic operation of the KR was observed due to many hundreds of Reads and Writes occurring during power-up.
3) Do not use Port 3 for accessing external memory while using the Slaveport.
4) One address line must be dedicated to select which register pair of the Slaveport is accessed.
5) The best way to debug code is to place the code in internal RAM. This allows the code to be modified to solve problems as they arise. The only other alternative is to take blank OTPs and program them. This alternative is a slow and expensive method.

```
Name        SLAVPAL1;
Partno      80C196KR;
Revision    01;
Date        2/21/92;
Designer    Bob Johnson;
Company     ;
Assembly    80C196KR AT Interface board;
Location    ;
Device      P22V10;
/*****************************************************************/
/* Generates CS# for KR on AT bus using address 210 to 211       */
/*****************************************************************/
/* Allowable Target Device Types:   CE22V10                      */
/*****************************************************************/

/** Inputs **/

PIN [2..10] = [a1..a9];       /* Address lines                   */
PIN 11 = !AEN;               /* 0 when CPU cycles are being used */

/** Outputs **/

PIN 19 = !CS;                /* 0V=> enable slaveport           */

/** Declarations and Intermediate Variable Definitions  **/

FIELD    memaddr = [a9..1];

/** Logic Equations **/

slaveport  = (memaddr:[210..211] & AEN);

!CS = !slaveport;
```

Appendix A. (22V10 program).

Designing with 3 Volt/5 Volt Mixed Voltage

John Williams
Applications Engineer
Intel

Introduction

One of the greatest difficulties in designing a low voltage system is product availability. Currently, there are a large number of manufacturers producing 3 Volt products (3 Volts refers to any low operating voltage device). Embedded processors, like the 80L186EA and EB have been available at 3 Volts for some time. Memory and logic are also widely available (notice there was no mention of short lead times). Other peripheral components are still not available at 3 Volts. Many of these components require redesign to operate at low voltages. Until these components are replaced with low voltage versions, systems requiring their services will have to utilize the 5 Volt version. This causes tremendous headaches for system designers. System cost, complexity and power consumption all suffer due to the lack of a complete selection of low voltage devices.

Interfacing 3 Volt and 5 Volt Components

Somewhere within a mixed voltage design, the designer must interface 3 Volt and 5 Volt devices. This can happen in 3 Volt to 5 Volt translation, 5 Volt to 3 Volt translation, bidirectional translation and 3 and 5 Volt devices residing on the same system bus. As simple as these interfaces may seem, there are still some problems to be solved.

3 Volt to 5 Volt Interface

One way to translate a 3 Volt output to a 5 Volt level is with ACT/HCT logic. These parts accept a TTL level input and give a CMOS output. When operating at 5 Volts, these parts see a 3 Volt level input the same as they would see a 5 Volt TTL level input. The output will be a 5 Volt CMOS level. This is a relatively straight-forward approach, but it does have the problem of additional current consumption (Icc-delta).

5 Volt to 3 Volt Interface

Unfortunately, a 5 Volt output cannot be connected directly to a 3 Volt input. Under the 'Absolute Maximum Ratings' section in most data sheets, there is a specification for the maximum voltage on any input pin. This number is typically Vcc + 0.5 Volts, and on some 3 Volt devices goes down to Vcc + 0.3 Volts. A 5 Volt CMOS or TTL output

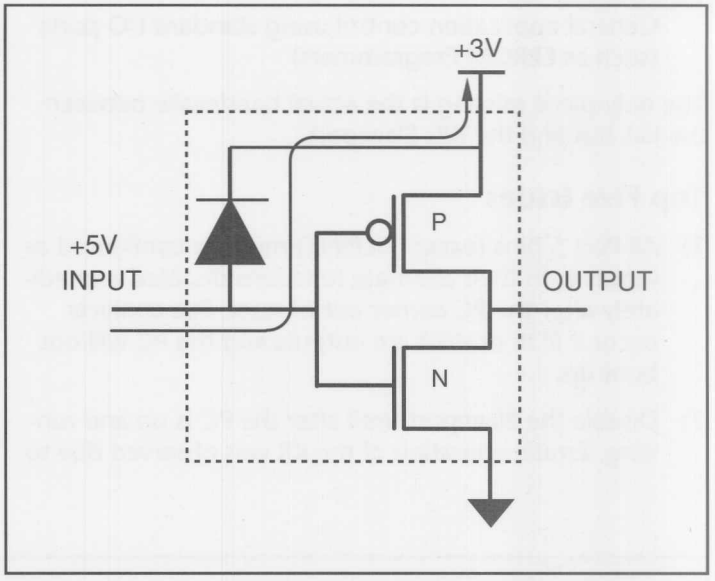


Figure 1. 5 Volt Output Driving a 3 Volt Input

high will typically drive close to Vcc. When the maximum input voltage is exceeded, the ESD protection diode on the input of the device will be forward biased and current will flow into the 3 Volt Vcc (Figure 1). This could draw the 3 Volt supply up to the input voltage minus the voltage drop across the ESD diode. Connecting a 5 Volt output to a 3 Volt input will lead to long term reliability problems. Device manufacturers are developing input buffers for low voltage devices that are 5 Volt tolerant, but until these devices are widely available, other solutions must be found.

There is a way to work around the problem of driving a 3 Volt input with a 5 Volt output. Series resistors can be placed on the outputs of the 5 Volt device. The goal is to drop the voltage to a level acceptable by the 3 Volt device. The value of the resistor must be chosen to limit the current into the 3 Volt device. The major sacrifice in this solution is speed. If the system can handle the speed degradation, then it may be a simple, inexpensive way to translate from 5 Volts to 3 Volts. As a final point on this subject, if this solution is implemented, considerations must be made for system power-up. If the 5 Volt supply ramps much faster than the 3 Volt supply, the substrate diode may still be forward biased temporarily, leading to reliability problems. This situation must be taken into account when determining the resistor value. Assume the worst case where the 5 Volt part is running at 5.5 Volts and the 3 Volt part has Vcc = 0 Volts.

Voltage Translation with Open Drain Outputs

Open drain output devices provide possibly this simplest way to convert from 3 Volts to 5 Volts and vice-versa. All that is required is an external pullup resistor to the desired output voltage. If the open drain device outputs a logic '1,' there is virtually no current consumption penalty for the conversion. If the output is a logic '0,' there is a cur-

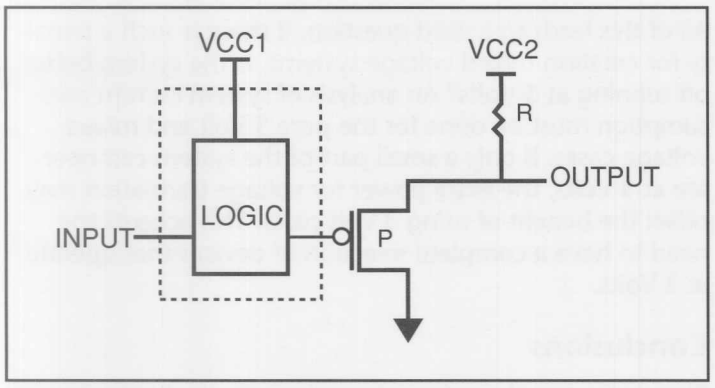


Figure 2. Voltage Translation with Open Drain Outputs

rent path to ground, but a high resistance pullup will limit the amount of current (at the cost of speed).

If an open drain device is not available, the function can be easily duplicated using an external MOSFET and a resistor (Figure 2). The output to be translated connects to the gate of the transistor. An n-transistor connected to ground with a pullup to Vcc will act as an inverter for the output. A p-transistor connected to ground with a pullup resistor will not invert the output.

Bidirectional Translation

The methods described above all work well for translating unidirectional signals. What happens if a 5 Volt peripheral resides on a 3 Volt bus? The only way to use any of the previous solutions would be to create a block of logic consisting of two back-to-back unidirectional translators. This is not a good solution for a designer attempting to minimize board real estate. Fortunately, a number of manufacturers are working on and producing buffers which have two Vcc pins. The devices translate bidirectionally between the two Vcc values. These are the most practical solution for bidirectional translation between 3 and 5 Volts.

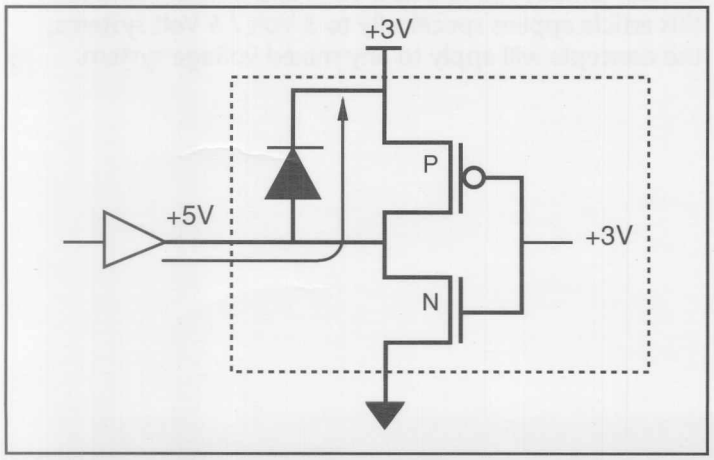


Figure 3. 5 Volt Device Driving a Floated 3 Volt Output

Mixed Voltages on the Same Bus

In an Ideal situation, a designer could place a 5 Volt device and a 3 Volt device on the same system bus. Unfortunately, a floated 3 Volt output will be damaged when a 5 Volt part drives the bus. Depending on the state of the inputs to the 3 Volt device output buffers, it is possible that the p-transistor will turn on (Figure 3). If this device turns on, the 5 Volt supply and 3 Volt supply will be shorted together. Even if this situation does not occur, the 5 Volt signal will still forward bias ESD protection diodes inside the 3 Volt device, creating a situation similar to a 5 Volt output driving a 3 Volt input. System busses should run at a single voltage with parts operating at other voltages being buffered.

Disadvantages of Mixed Voltage Systems

It may appear obvious by this point that there really are no advantages to mixed voltage systems. They only exist because of the absence of a complete selection of low voltage devices. During the transition of the industry to 3 Volts, the object of designers is to minimize the disadvantages of having a mixed voltage system. Two of the major drawbacks of mixed voltage systems are: voltage supply requirements and additional current consumption.

Multiple Supplies in Mixed Voltage Systems

A major disadvantage of mixed voltage systems is the requirement of multiple voltage supplies. A typical system may require 3 Volts (major components, memory), 5 Volts (older peripherals, small displays), +/- 12 Volts (RS-232 communications) and even higher voltages (backlit VGA displays, etc.). One goal in designing a mixed voltage system is to minimize the number of required voltages and the number of devices used to create them. To avoid the extra chip-count associated with creating multiple voltages, some manufacturers, Maxim for example, offer one-chip solutions to provide 3V, 5V, and a third variable output from 2 or 3 cells. The designer can also take advantage of parts with internal charge pumps that take 3 Volt or 5 Volt inputs and create the output voltage levels they require. Although multiple voltages can easily be created with a minimal number of chips, the designer still pays the price for having non-3 Volt parts in the system: battery life. As technology moves forward and 3 Volt designs gain momentum, more components will be available at low voltage (3 Volts or less). This will eliminate the requirement for multiple system voltages and the added system cost and complexity associated with creating them.

Additional Current Consumption in Mixed Voltage Systems

Interfacing 3 Volt and 5 Volt devices in a mixed voltage system is unavoidable. Regardless of how a designer

implements these interfaces, they will all have one common characteristic, additional current consumption.

Some devices have an additional specification called Iccd (or Icc-delta). This specification defines the additional current consumed, per input pin, if an input high voltage is equal to Vcc - 2.1 Volts. This situation closely resembles using ACT or HCT logic for 3 Volt to 5 Volt translation. This number can be up to 1.5 mA per input pin. Consider a unidirectional, 16-bit bus translated from 3 Volts to 5 Volts using ACT logic. In a worst case situation, this can be a major source of continuous current consumption. This is a maximum value, though, typically the extra current will amount to about 100 to 200 µA per input pin. Additionally, this specification only applies to a logic '1' input, and in most situations only a fraction of the 3 Volt inputs will be high at any one time.

The reason behind the extra current consumption when using ACT/HCT logic for 3 Volt to 5 Volt translation lies in the input buffers (Figure 4). If the input of the device is driven all the way to Vcc, the p-transistor will be completely off and the n-transistor will be completely on. This can be modeled as 5 Volts connected to ground through a 5 MOhm resistor. As shown by the graph in Figure 4, the only current flowing will be leakage current, almost nothing. As the voltage on the input moves farther away from Vcc, the input transistors move closer to their saturation

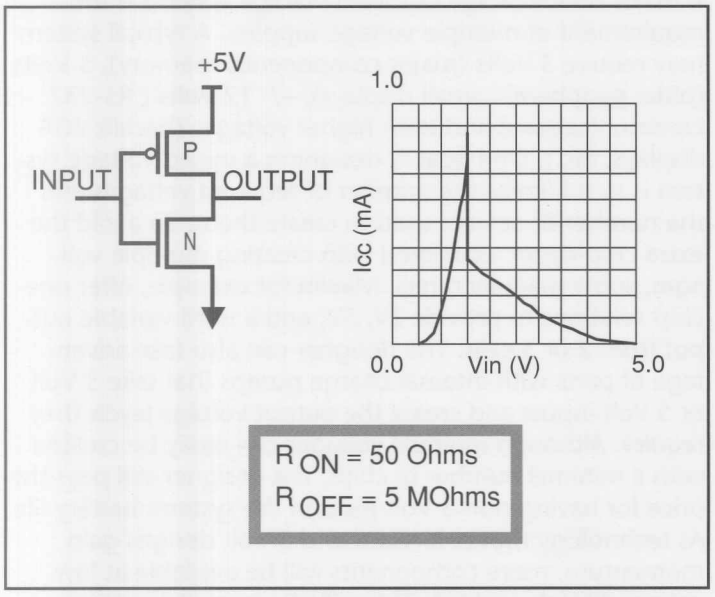


Figure 4. Current Consumption When Using ACT/HCT Logic for Voltage Conversion

region. The resistance path through the transistors to ground will decrease from the initial 5 MOhm. This will increase the current flow through them. The graph shows this current to be on the order or 150 µA per input (at room temperature).

All of this leads to a valid question. If there is such a penalty for creating mixed voltage systems, is the system better off running at 5 Volts? An analysis of system current consumption must be done for the pure 5 Volt and mixed voltage cases. If only a small part of the system can operate at 3 Volts, the extra power for voltage translation may offset the benefit of using 3 Volt parts. This accents the need to have a complete selection of devices that operate at 3 Volts.

Conclusions

There are a number of considerations that a designer must make when designing a mixed voltage system. Interfacing 3 Volt and 5 Volt logic must be done carefully. There are a number of solutions to do this, but if done incorrectly, the system can eventually fail. Many manufacturers provide simple solutions to do unidirectional and bidirectional transfers. These solutions are probably the easiest to implement and consume a minimal amount of power.

A mixed voltage system, by definition, consumes more current than a pure 3 Volt system. This added current consumption comes from different sources. Any method used to translate from one voltage to another requires additional current. A mixed voltage system also has more devices than a pure 3 Volt system. Extra devices are needed for voltage translation and creating required system voltages. In addition to drawing current, these extra devices increase system size, cost and complexity.

As technology advances and manufacturers redesign current parts, complete systems will be able to operate at 3 Volts. Until that time, mixed voltage systems will have to exist. Although a mixed voltage system requires more power than an entirely 3 Volt version, it still consumes less power than a 5 Volt version. Mixed voltage systems will exist in some form for a long time. Right now, they exist because of the conversion from 5 Volt devices to 3 Volt devices. Mixed voltage systems will continue to appear as industry takes the steps to lower operating voltages. Some devices already operate at 2 Volts and below. Although this article applies specifically to 3 Volt / 5 Volt systems, the concepts will apply to any mixed voltage system.

Operation of Powerdown in the 80C196 Devices

Gary Harris
Applications Engineer
Intel

Introduction

The powerdown feature allows the KB,KC and KD devices to be placed into a very low power state while retaining all information in register RAM. All peripherals are powered down, retaining their current state, and the oscillator is shut off. Power consumption drops into the microwatt region (refer to data sheet for exact specifications).

Entering Powerdown

The powerdown feature is enabled if CCR.0 = 1. Powerdown mode is entered by executing the IDLPD #2 instruction. Before this is done, the following housekeeping is good practice:

1. Serial port transmissions and receptions should be allowed to complete. Otherwise, when powerdown is exited the activity will continue where it left off and may result in incorrect data being transferred.
2. Analog conversions should be allowed to complete or else an incorrect result will be returned.

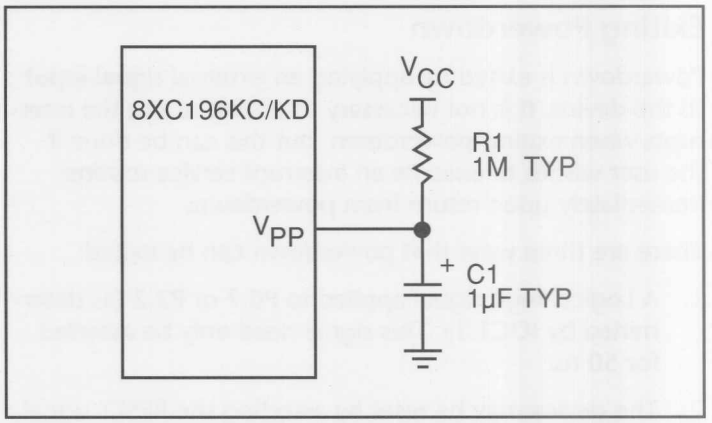


Figure 1. External Connection for V_{PP}

3. If the watchdog timer (WDT) is being used, it should be cleared immediately prior to entering powerdown. This ensures there is enough time to service the WDT when returning from powerdown.

Note that in order for the part to enter powerdown, the pin which is configured as EXTINT (P0.7 or P2.2) must be held low.

During powerdown, the internal oscillator and clock generator is disabled. If the device has access to internal OTPROM (EA# = LOW) ports 3 and 4 are open drain and hold the last value output. If the device was set for external execution only, (EA# = HIGH), ports 3 and 4 will float. All other pins remain in the state the last held prior to entering powerdown.

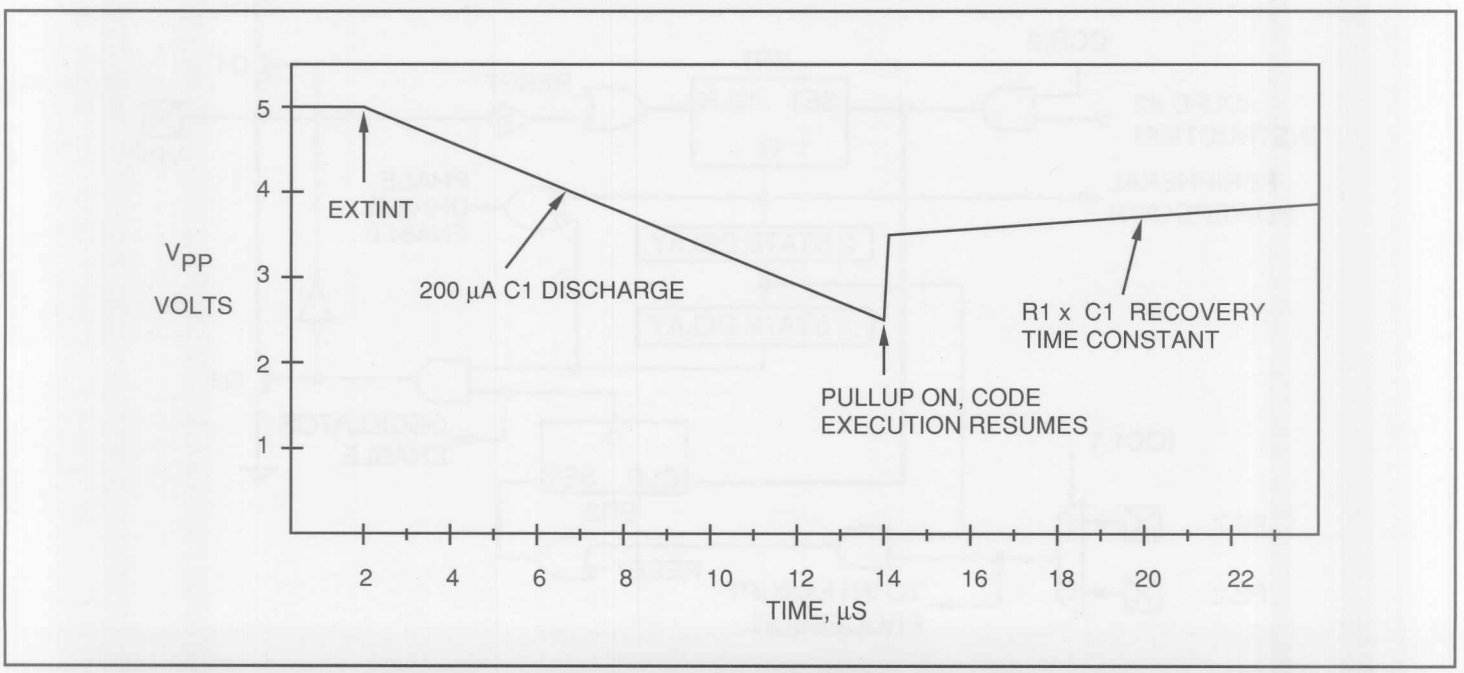


Figure 2. Typical V_{PP} Voltage During Return from Powerdown

Exiting Powerdown

Powerdown is exited by applying an external signal input to the device. It is not necessary to enable or use the interrupts when exiting powerdown, but this can be done if the user wishes to execute an interrupt service routine immediately upon return from powerdown.

There are three ways that powerdown can be exited:

1. A Logical high signal applied to P0.7 or P2.2 (as determined by IOC1.1). This signal need only be asserted for 50 ns.
2. The device may be reset by asserting the RESET signal. If this is done, RESET must remain asserted until the oscillator has stabilized. If an external oscillator is being used, RESET only needs to be asserted for 1 state time.
3. If an external clock is being used, driving the VPP pin low for at least 50 ns will immediately bring the part out of powerdown. This method provides the fastest return from powerdown.

When using P0.7 or P2.2 to generate an interrupt upon to exiting powerdown, These signals must remain high for at least 2 state times after CLKOUT has resumed.

Circuit Recommendations

Figure 1 is the recommended circuit for exiting powerdown using P0.7 or P2.2. Figure 2. shows the voltage verses time characteristic for VPP. The circuitry is designed to generate a time delay before turning on the internal oscillator. This allows the oscillator time to start-up and stabilize. This delay is generated by discharging capacitor C1 connected from the VPP pin to ground. When the P0.7/P2.2 input is asserted, the capacitor is discharged at a typical rate of 100 to 200 µA until it reaches a threshold voltage of about 2.5 volts. The time for this to occur can be calculated as follows:

$$\text{TIME} = \frac{C \times V}{I}$$

For a capacitor value of 1 uF, a ΔV of 2.5 volts and assuming 200 µA discharge current, the time will be 12.5 ms.

When the threshold voltage is reached, a pull-up transistor is turned on which quickly pulls the pin up to about 3.5 volts. At this point the pullup becomes ineffective, and the external resistor R1 completes the pull-up to VCC ("recovery" time, figure 2). The time constant for recovery follows an exponential charging curve, and for C1 = 1 uF and R1 = 1 Megohm, this will take several seconds.

If powerdown is re-entered before C1 charges to VCC, it will take less time for the voltage to ramp down to the threshold and exit powerdown (because ΔV is less). This should be taken into account when allowing for worst-case oscillator start-up times.

Recommended Values for R1 and C1

R1 and C1 are not critical. R1 should be kept large so that it does not interfere with the discharge current (about 200

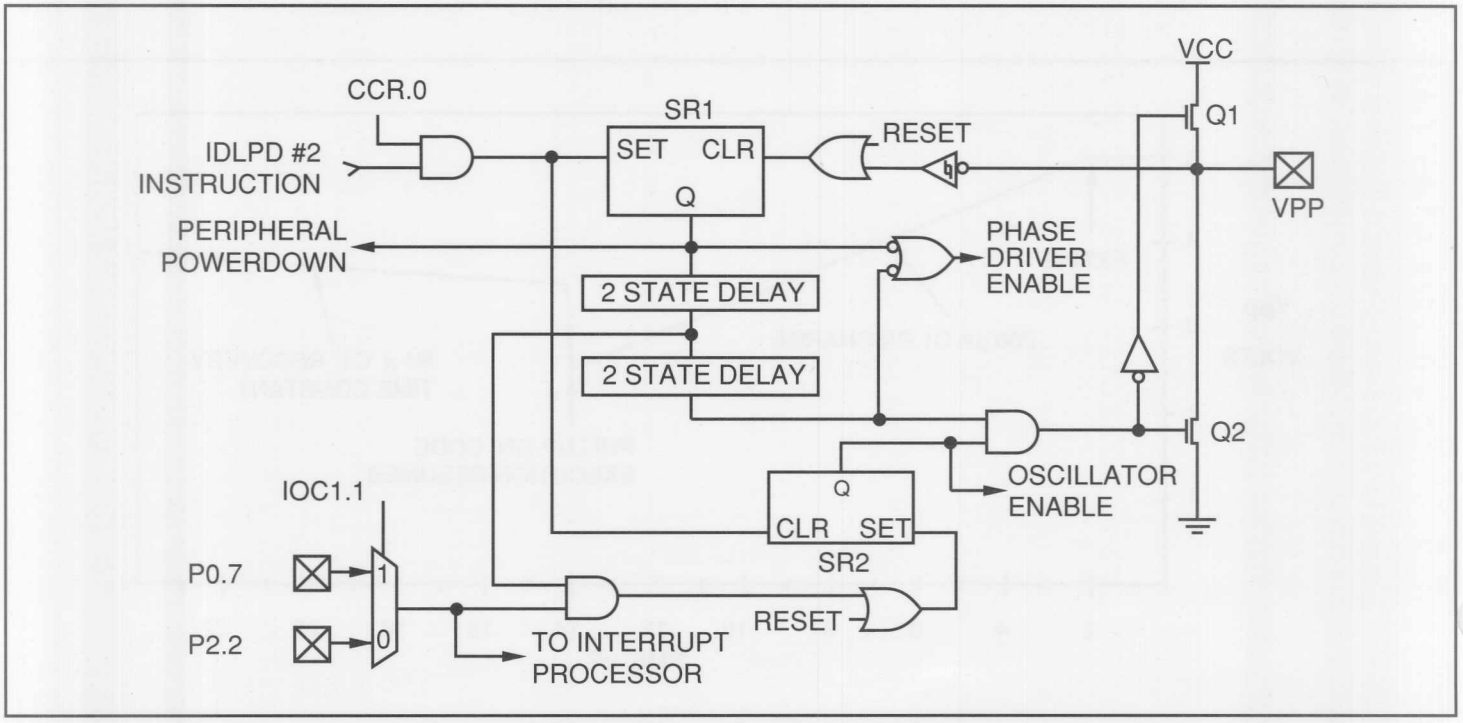


Figure 3. Internal Powerdown Control

μA). Values between 200K to 1 Megohm should perform satisfactorily. C1 should be selected to provide the worst-case time constant needed for the oscillator to stabilize. All designs should be characterized under worst-case conditions to verify proper operation.

If an external oscillator is used, the value of C1 can be very small to allow rapid recovery from powerdown. A 100 pF capacitor is acceptable, and will result in a 1.25 us discharge time.

If an external clock is used, and powerdown is exited by pulling VPP low, R1 and C1 are not needed.

Description of Powerdown Control Circuitry

Figure 3 shows the internal circuitry used to control powerdown.

Definition of terms:

1. IDLPD #2. This signal is generated when the IDLPD #2 instruction is executed, and causes the device to enter the powerdown state.
2. Phase Drive Enable. This signal enables the internal phase clocks of the device.
3. Oscillator Enable. This signal enables the internal oscillator circuitry.
4. Peripheral Powerdown. This signal is used by the peripherals to enter the powerdown state.

Operation

When the device is reset, SR1 is cleared and SR2 is set. This enables the phase clocks and internal oscillator. The pullup (Q1) will be on and the pulldown (Q2) will be off. This is the normal operating mode. When an IDLPD #2 instruction is executed, SR1 is set and SR2 is cleared. This disables the phase clocks and oscillator. This also enables the P0.7/P2.2 input to SR2.

When P0.7/P2.2 is asserted, SR2 is set, re-enabling the oscillator and turning on the weak pulldown Q2. This begins discharging the external capacitor connected to VPP. When the voltage drops below the threshold voltage, the Schmitt trigger clears SR1, enabling the phase clocks, pull-up Q1, and disabling Q2.

Glad you asked...

Ports 3/4 and Address/Data Bus Operation for the 8X9XBH, 8XC196KB, 8XC196KC, and 8XC196KD

Richard N. Evans
16-Bit Applications Engineer
Intel

There have been a few questions on the operation of the address/data bus in BUS MODE and PORT 3/4 MODE. Following is an explanation of how the Ports or bus behave in these modes. Two cases to consider are:

Case 1: EA# Tied Low (External Execution)

- The bus is strongly driven during bus cycles.
- Reading and writing Ports 3/4 (1FFEH) causes a bus cycle with the bus strongly driven just like any other memory access.
- When the bus is idle it floats regardless of the last value written to Ports 3/4.

Case 2: EA# Tied High (Internal Execution)

- Reading and writing to Ports 3/4 (1FFEH) does not cause a bus cycle. Instead Ports 3/4 become open drain I/O Ports.
- If an external memory location is accessed, a bus cycle occurs. After the access, the Port 3/4 is returned to an open drain I/O Port.
- When used as I/O Port pullups to VCC are needed (10K).

Understanding Internal Chip Grounding On 87C196Kx Family Microcontrollers

Chris Banyai
Applications Engineer
Intel

Overview

The KX family includes the 8XC196KR, JR, KQ, JQ and 8XC196KT, JT advanced 16-bit CHMOS microcontrollers. These are highly integrated, high performance microcontrollers that are excellently suited for automotive applications such as ABS brakes. In the harsh automotive environment, to get the best performance it is helpful to understand the internal relationship between the different ground pins located on these devices. Included in this document is a description of the ground pins found in the KX family, a description of the KX internal ground regions, explanation of parasitic device turn-on by differing ground potentials, and a section addressing the important applications considerations for a robust KX microcontroller grounding scheme.

Device Ground Pins

On KX microcontrollers four ground pins supply the three major ground regions which are the core, ports, and analog circuitry. Partitioning the grounds internally provides noise isolation. Without this scheme, noise sources such as port driver switching could directly couple into the CPU, analog to digital converter, or other logic deeper within the microcontroller. One Vss pin supplies the core region including the CPU, two Vss1 pins supply the port region, and the ANGND pin supplies the analog region including port 0. In addition to the Vcc and Vref (Analog Reference Voltage) pins, all four ground pins must be connected for the device to function properly.

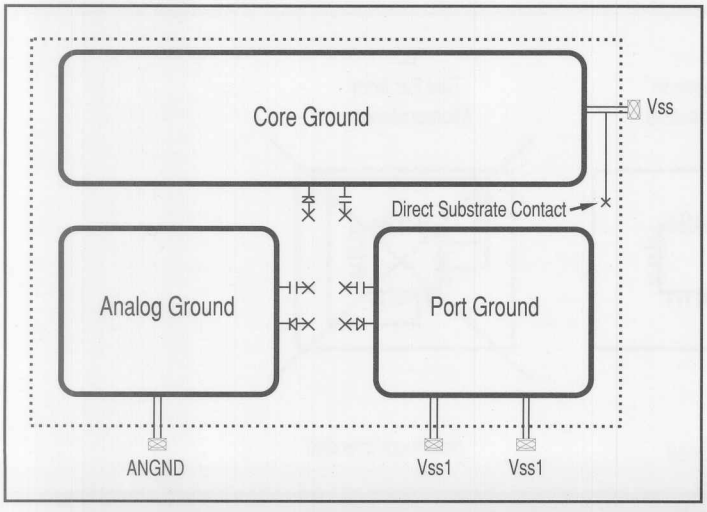
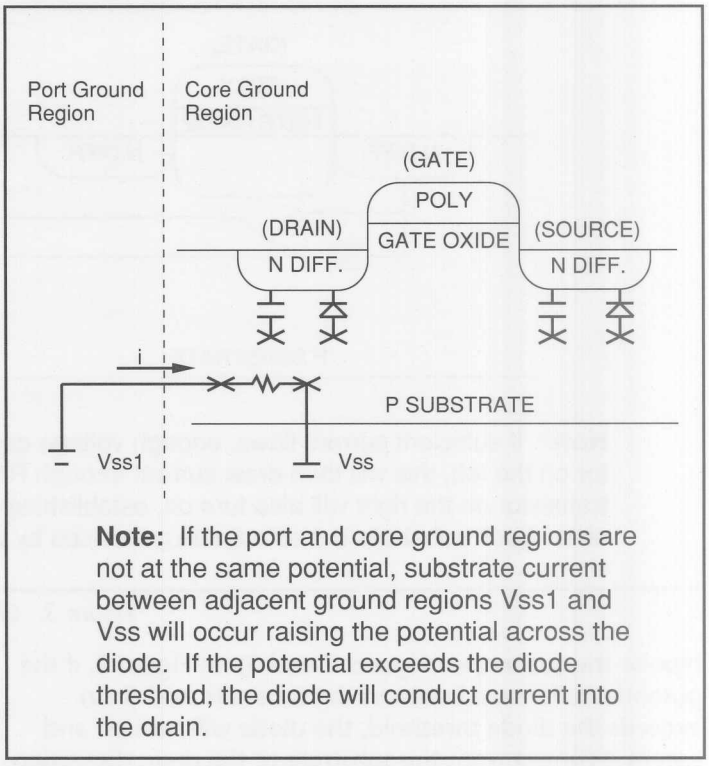


Figure 1. KX Ground Regions



Note: If the port and core ground regions are not at the same potential, substrate current between adjacent ground regions Vss1 and Vss will occur raising the potential across the diode. If the potential exceeds the diode threshold, the diode will conduct current into the drain.

Figure 2. Parasitic Device Turn-on

VREF Connections Count

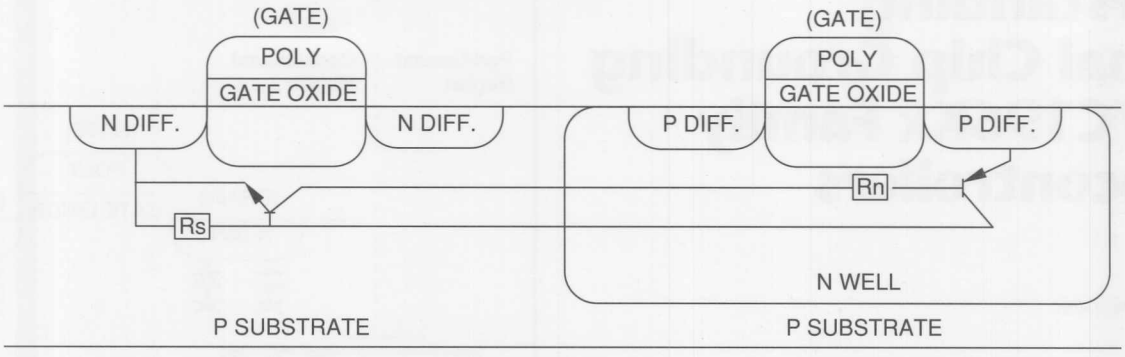
Vref must be connected even if the on-chip A/D converter is not being used because these pins supply the digital portion of Port 0 as well as the analog section of Port 0.The Vss pins used for port ground (Vss1) are pins #68 and #30 on 68 lead devices and pins #1 and #25 on 52 lead devices. The Vss pin used for core ground (Vss) is pin #5 on 68 lead devices and pin #3 on 52 lead devices.

KX GROUND Regions and Associated Pins

The three noise isolating ground regions on the KX family are: core, port, and analog ground. Core ground is supplied by the Vss pin, port ground is supplied by two Vss1 pins, and analog ground is supplied by the ANGND pin. Each of the ground regions is coupled to the substrate and therefore to each other by parasitic devices as is denoted in Figure 1 by a capacitor and diode connected to the substrate (X).

Parasitic Device Turn-On by Differing Ground Potentials

Parasitic devices are formed at the PN junctions of the CMOS transistors and consist of capacitors, diodes, and



Note: If sufficient current flows, enough voltage can be generated across Rs to turn on the parasitic transistor on the left, this will then draw current through Rn and, if the voltage developed is sufficient, the parasitic transistor on the right will also turn on, establishing a self-sustaining low resistance path between the supply rails. Such substrate current can be generated by potential differences between ground regions.

Figure 3. CHMOS Latch-up

bipolar transistors (see Figures 2 and 3). In Figure 2, if the potential difference between the substrate and drain exceeds the diode threshold, the diode will turn on and conduct current from the substrate to the drain disrupting the desired drain voltage. These diodes form the base-emitter structure of parasitic bipolar transistors responsible for latch-up, as in Figure 3.

Design Considerations for Robust Grounding

A robust grounding scheme provides noise-free stable ground, avoids the activation of parasitic devices, minimizes the coupling of noise across ground regions (noise isolation), and eliminates power consumption from ground loop leakage. The term robust refers to a reliable grounding scheme despite a device and electrical environment variation. The key application design considerations for a robust grounding scheme are: maintenance of all ground pins nominally at the same potential (within 50mV); and, the provision of a low impedance path to ground for each ground pin.

Maintain Ground Pins at A Common Potential

The recommended method for maintaining all ground pins at the same potential is to connect Vss, Vss1, and ANGND together via "taps" into a ground plane located within the circuit board. By this method all three ground regions (core, port, analog) are maintained at the same potential avoiding the activation of parasitic devices and possible latch-up. The partitioning of the chip into three separate ground regions minimizes the coupling of noise across different parts of the chip. Since each pin is at the same potential no current will flow through the substrate between two ground pins such as Vss and Vss1.

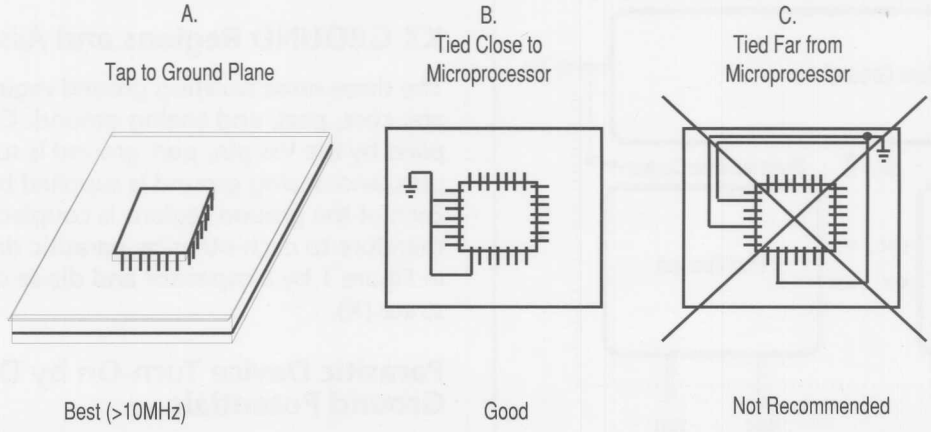


Figure 4. Low Impedance Grounding Schemes

Low Impedance Paths

Providing a low impedance path to ground for each ground pin is also important for a robust grounding scheme. This characteristic is intrinsic to the recommended "tap to ground plane" method. Another scheme, while not as effective as the "tap to ground plane" method, is to connect the ground pins to a common reference node as close to the microcontroller as possible.

Long Trace Effects

Avoid connecting several long traces at a common point across the circuit board. Long traces add impedance. When several circuits share a common ground wire, currents from one circuit, flowing through the finite impedance of the common ground line, cause variations in the ground potential of the other circuits. Given, that the currents in a digital system tend to be spiked, and that the common impedance is mainly inductive reactance, the variations could be bad enough to cause bit errors in high current or particularly noisy situations. The best strategy is to provide a low impedance path to ground.

8XC196Kx Family TIJMP Instruction

By David Boehmer
Applications Engineer
Intel

Intel Applications Engineering often receives questions regarding the TIJMP Instruction. The following text is intended to clarify TIJMP instruction execution and its use in your code.

First of all, a little housekeeping is in order. Errata #5 in the 1992 87C196KR Automotive data sheet lists the C-step device as having a problem which does not exist. Please disregard errata #5 in the 1992 Automotive KR data sheet. This error will be corrected in future revisions. Sorry for any inconvenience. Let's look at the TIJMP instruction code format and define the various operands used with the instruction.

Instruction format:

TIJMP tbase,[index],#index_mask

tbase: A word register containing the 16-bit base address of the jump table. tbase may be located in RAM up to 0FEh without using windowing or it can be located in RAM above 0FFh by using windowing. The jump table may be placed at any non-reserved memory location on an even word boundary.

index: Word register containing a 16-bit address that points to a register. The register pointed to contains a 7-bit value used by the TIJMP instruction to calculate the offset into the jump table.

Index may be located in RAM up to 0FEh without using windowing or it may be located in RAM above 0FFh by using windowing. The 16-bit address contained in index is absolute and disregards any windowing in effect at the time TIJMP executes.

index_mask: An 7-bit immediate data mask for the index register. index_mask is ANDed with the 7-bit value pointed to by index. It is then multiplied by two to determine the correct offset into the jump table.

offset = [index] .and. index_mask

destination = (2 * offset) + tbase

> **Note:** The multiply by two provides for word alignment of the jump table. This needs to be taken into consideration when using the TIJMP instruction for EPAIPV vector decoding and jump table initialization.

Example Using TIJMP to Decode the EPAIPV Vector

Before we take a look at the accompanying code example, it is important to understand the EPAIPV register operation. When read, this register returns a value that corresponds to the highest priority, non-masked, pending indirect interrupt. This value is used by the TIJMP instruction

```
EPAINTx_ISR:    PUSHA                                   ;saveINT_MASK/INT_MASK1/WSR/PSW
                LD      JTBASE_PTR,#JTBASE              ;store jump table base address
                LD      EPAIPV_PTR,#EPAIPV              ;read EPAIPV offset
                TIJMP   JTBASE_PTR,[EPAIPV_PTR],#7Fh    ;initiate jump to correct ISR
OVR_EPA0_ISR:   .                                       ;EPA0 over-run routine
                .                                       ;
                .                                       ;
                TIJMP   JTBASE_PTR,[EPAIPV_PTR],#7Fh    ;check for pending interrupts, exit
                                                        ;
EPAINTx_DONE:   POPA                                    ;
                RET                                     ;exit, all EPAINTx interrupts srvc'd.

JTBASE: DCW     EPAINTx_DONE                            ;0 (no interrupt pending)
        DCW     OVR_TMR2_ISR                            ;1 (Timer2 overflow)
        DCW     OVR_TMR1_ISR                            ;2 (Timer1 overflow)
        DCW     .                                       ;
        DCW     .                                       ;
        DCW     .                                       ;
        DCW     OVR_EPA0_ISR                            ;0Eh    (EPA0 overflow)
        DCW     .                                       ;
        DCW     .                                       ;
        .       .
```

as an offset to fetch a word from a jump table (JTBASE in this example) which contains start addresses of interrupt service routines. On KR A-step devices, the value returned when reading EPAIPV will range from 00h to 28h. On C-step devices the value returned will range from 00h to 14h. Compared to C-step devices, A-step devices in effect multiply this value by two. This is covered in more detail in the following (Page 20) article, "8XC196KR A-1 to C-Step EPAIPV Design Considerations".

Referring to the code example, assume that an OVRINT0 indirect interrupt occurs on EPA0. This interrupt signals that an edge has been detected on this channel and that both the input buffer and EPA_TIME0 are full. Assuming that EPAINTx is enabled, OVR_EPA0 is enabled, and interrupts are globally enabled, the EPAINTx_ISR interrupt service routine will be entered.

Within EPAINTx_ISR the address of the jump table base and the address of EPAIPV are loaded into pointers. The TIJMP instruction is then executed and the OVR_EPA0_ISR interrupt service routine is entered. Note that instead of a RET instruction at the end of OVR_EPA0_ISR we use another TIJMP instruction. This is used to check for any other indirect interrupts that are pending. If EPAIPV contains a value of zero (no pending indirect interrupts) the EPAINTx_DONE interrupt service routine will be vectored to and a RET will be executed. This is to assure that EPAIPV is cleared before returning from the EPAINTx_ISR.

It is important to note that the accompanying code example is configured for KR C-Step devices where reading EPAIPV will return a value of 00h to 14h.

8XC196KR A-1 to C-STEP EPAIPV Design Considerations

By Dave Boehmer
Applications Engineer
Intel

A design change that occurred between the 8XC196KR A-1 and C-steps leaves a possibility of a code incompatibility between the two steppings. Let's have a look at the change that took place and what impact it will have upon applications switching from A-1 to C-step.

Description of EPAIPV

The Event Processor Array Interrupt Priority Vector (EPAIPV) is a byte register located at 1FA8h on the 8XC196KR. When read, this register will return a hexadecimal number in the range of 2h to 28h on the A-1 step (1h to 14h on the C-step). This corresponds to the highest priority interrupt source currently active. The EPAIPV will contain a zero if no interrupt source is active. The EPAIPV is commonly used in conjunction with the TIJMP instruction to decode the EPAINTX interrupt (20 possible sources). The TIJMP instruction uses the EPAIPV as an offset into a jump table.

A-1 vs C-STEP Operation

On the 87C196KR A-1 step, both EPAIPV and TIJMP multiply the vector value by two, which has a net effect of a multiply by four. In order for the jump table to be spaced properly, the user must either divide the EPAIPV by two (shift right one bit), enter word spaces between jump table vectors, or interleave the tables. The C-step was changed to allow the user to more fully optimize code and memory. On the C-step, only TIJMP will multiply the vector value. This C-step change results in correct jump table spacing.

CODE Compatibility Issue

This stepping change creates a problem for A-step code used on C-step devices. The problem occurs when the vector value returned by the EPAIPV is used as an offset to jump into the jump table. Depending upon how the user initially dealt with the A-step operation, there are three ways to implement a code change on C-step devices. Here are the descriptions:

1. If the user shifted the EPAIPV value one bit to the right on the A-1 step, the shift right instruction can simply be removed on the C-step devices. This is probably the easiest workaround.
2. If the user inserted word spaces between vectors in the jump table on the A-1 step, condense the jump table to include no spaces between vectors.
3. If the user interleaved tables on the A-1 step, the user must either shift the EPAIPV vector one bit to the left (essentially recreating the A-1 step situation) or relocate the base address of the second table to another location within the same code space.

Paged Addressed EPROM Plus 8051

The 8051 microcontroller provides two 64K memory blocks; one for program memory and one for data storage. Microcontrollers are generally used in program intensive environments which require large amounts of nonvolatile memory. Systems requiring up to 64K of memory can use EPROMs with linear addressing schemes, such as the 27512. For systems that require more than the 64K linear address space, the paged EPROM can add a new dimension to the 8051's program memory. Up to four of these devices can be used to provide 256 kilobytes (27513) or 512 kilobytes (27011) of executable memory without intruding on the data memory's space.

The 8051's hardware and instruction set facilitate easy read access of the program memory area but do not provide a simple means for writing to that area in order to change pages. The WR# output from port 3 of the 8051 could be routed directly to the WE# input of a paged EPROM to allow page changes whenever a write instruction addresses a location within a paged EPROM's memory partition. However, if four of these devices are used, filling all of the program memory space, the entire data area will be overlayed by the program memory area for all write instructions. Advantage can be taken of the output ports provided by the 8051 by designating a single bit of port 1, for example, to serve as a selector to allow the WR# output of the 8051 to be directed to either data memory or program memory. Figure 1 shows how the inclusion of simple logic gates can provide up to 512K of program memory and a full array of data memory.

For more information refer to AP-284 "Using Page-Addressed EPROMs" in the Memory handbook.

This article was reprinted from Appendix A of AP-284.

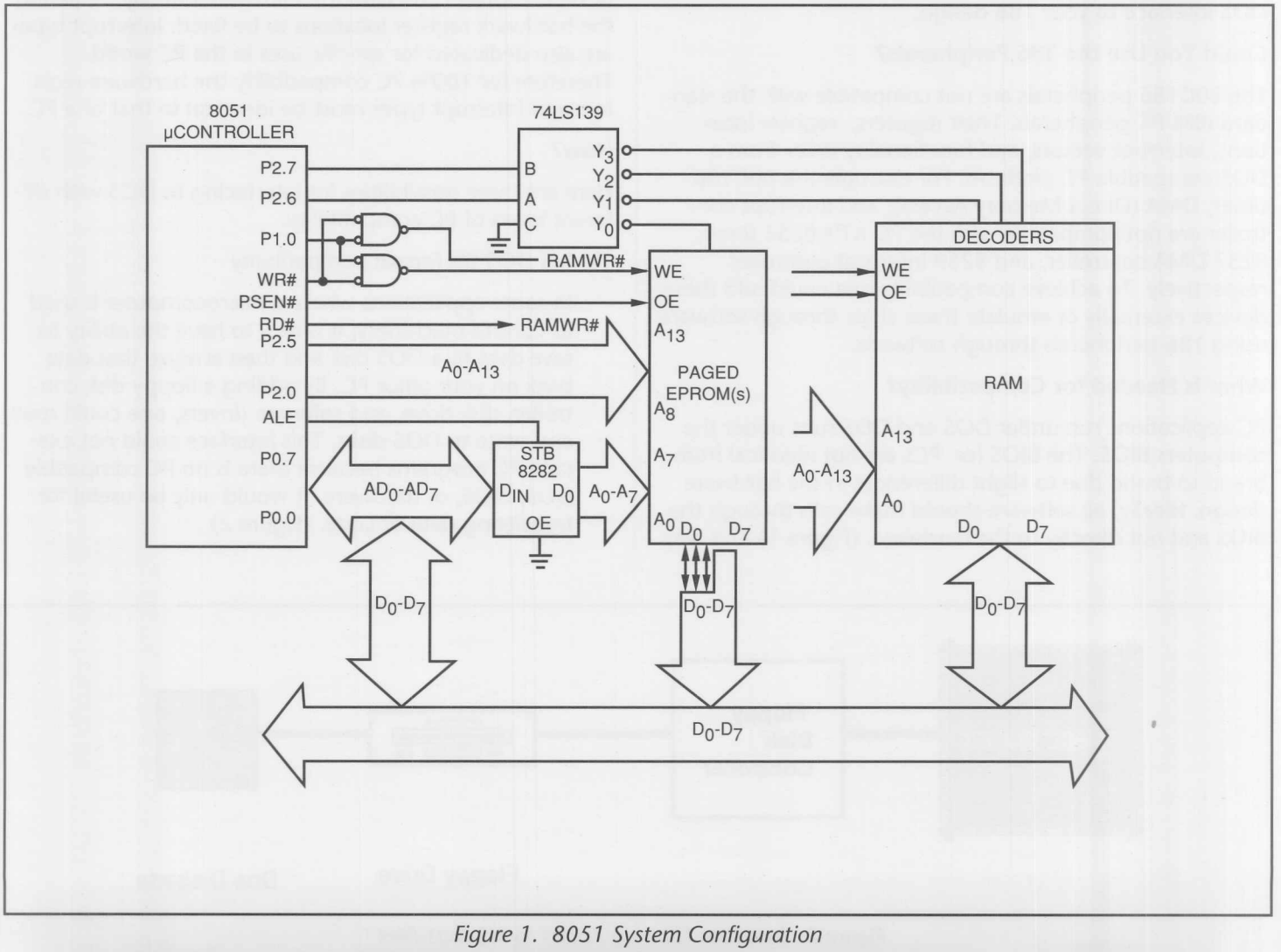


Figure 1. 8051 System Configuration

Interfacing DOS Using the 80C186 Family

Eric Auzas
Applications Engineer
Intel

Why?

Most 80C186 based designs are for control applications. Many of these systems could benefit from some level of DOS (Disk Operating System) interfacing. For example, writing or reading data to an MS DOS diskette for remote data analysis on a PC.

Development time could also be cut down if the system was BIOS (Basic Input Output System) or hardware level compatible, allowing DOS to execute. An 80C186 design could be upgraded from a system controller to also being PC compatible, thus allowing access to the PC software market. These are just some of the reasons for adding a DOS interface to your 186 design.

Could You Use the 186 Peripherals?

The 80C186 peripherals are not compatible with the standard IBM PC peripherals. Their registers, register locations, interrupt vectors, and functionality differ from a DOS compatible PC platform. For example the 80C186 timer, DMA (Direct Memory Access), and interrupt controller are not compatible with the PC XT's 8253 timer, 8237 DMA controller, and 8259 interrupt controller respectively. To achieve compatibility one could add these devices externally or emulate these chips through software using 186 peripherals through software.

What is Needed for Compatibility?

PC applications run under DOS and DOS runs under the computers BIOS. The BIOS for PCs are not identical from brand to brand due to slight differences in the hardware design. Ideally, all software should make calls through the BIOS and not directly to the hardware. (Figure 1) This way

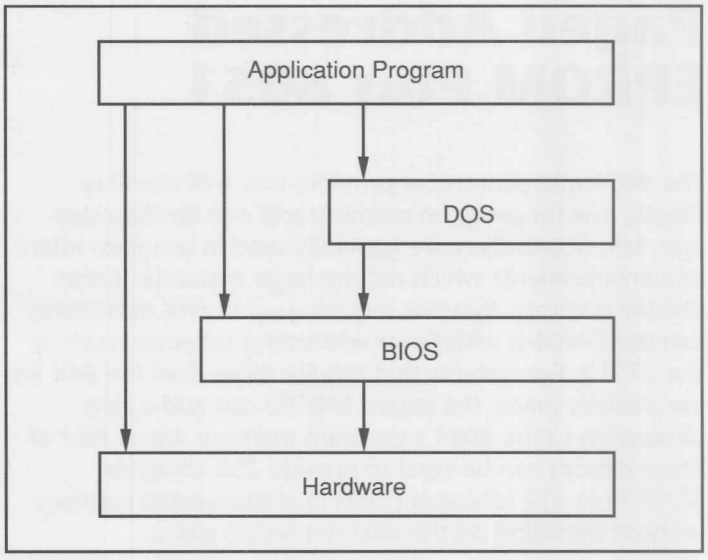


Figure 1. Three-layer-model

the hardware can change and the applications will never know the difference. In reality, PC software bypasses DOS or BIOS routines and calls the hardware directly, requiring the hardware register locations to be fixed. Interrupt types are also dedicated for specific uses in the PC world. Therefore for 100% PC compatibility the hardware registers and interrupt types must be identical to that of a PC.

How?

Here are three possibilities for interfacing to DOS with different levels of PC compatibility.

1) MS DOS file format compatibility

   In some applications where a microcomputer is used to control machinery, it is nice to have the ability to save data to a DOS disk and then analyze that data back on your office PC. By adding a floppy disk controller, disk drive, and software drivers, one could read and write to DOS disks. This interface could not execute PC programs because there is no PC compatible DOS, BIOS, or hardware. It would only be useful for transfering data or code. (Figure 2)

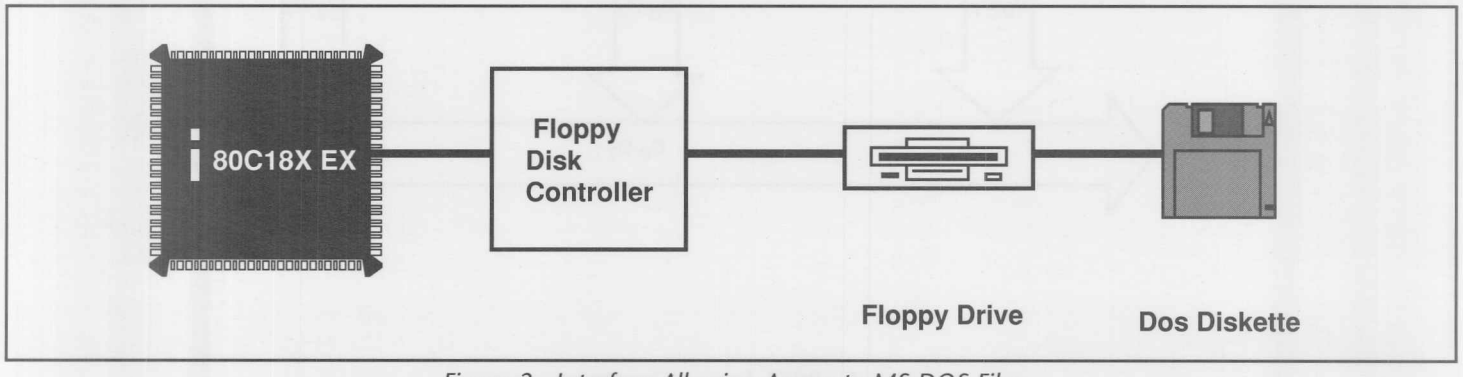


Figure 2. Interface Allowing Access to MS DOS Files

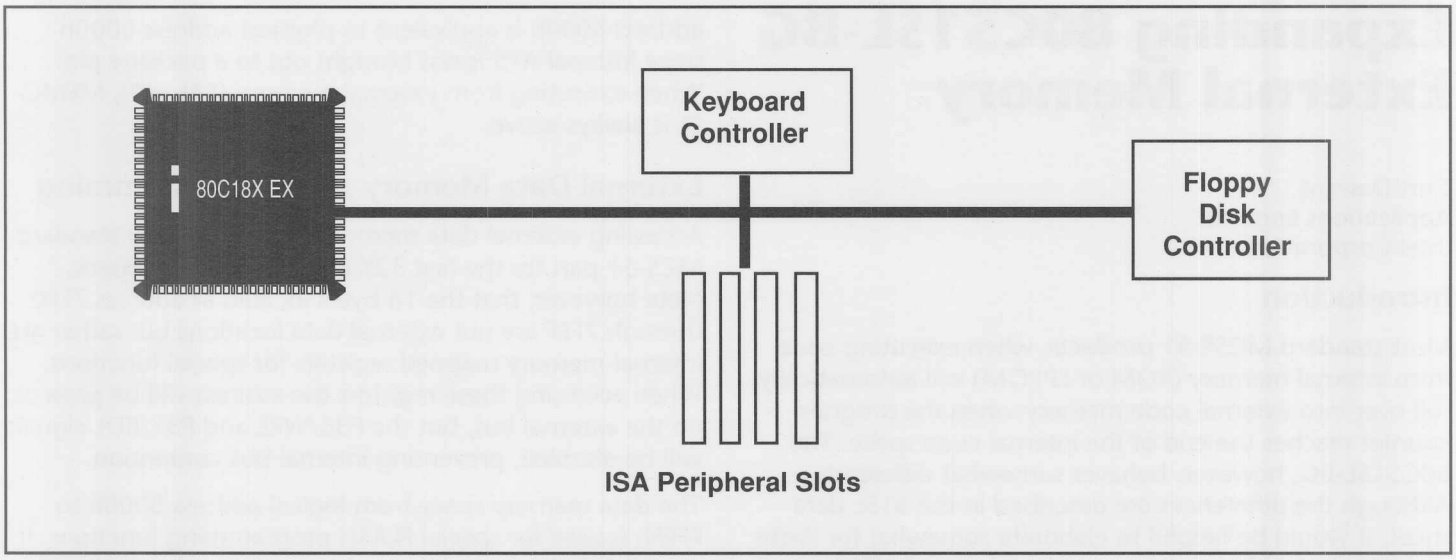


Figure 3. PC Bios Interface for Well Behaved Programs.

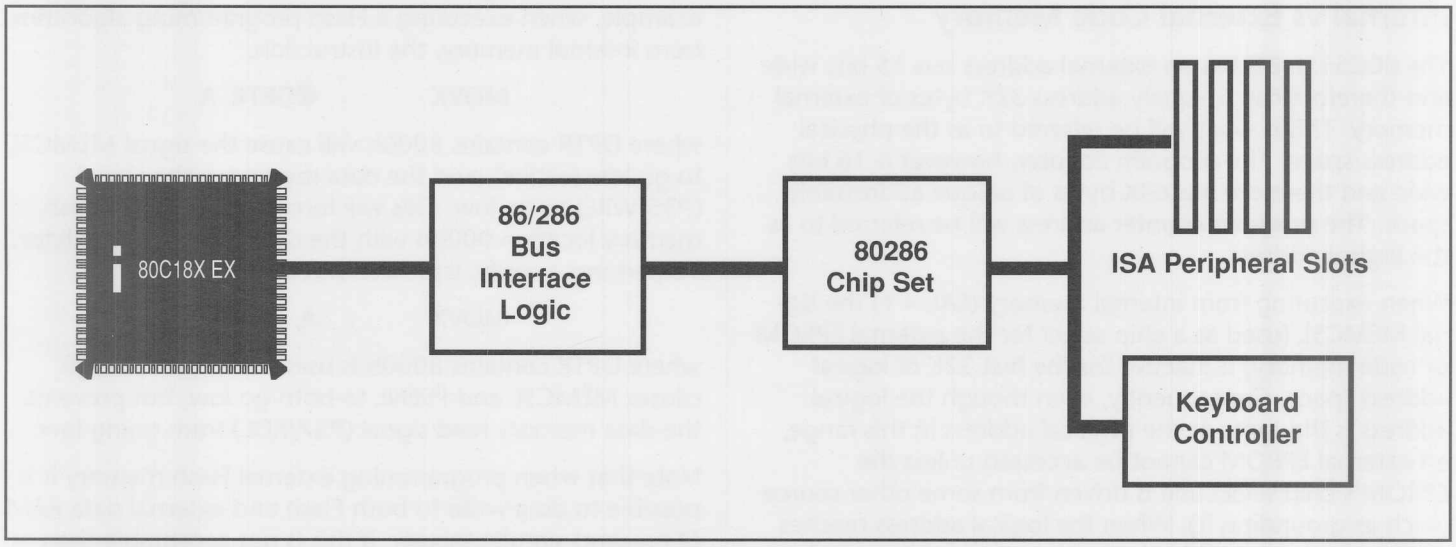


Figure 4. 100% Compatable PC Interface

2) BIOS compatibility

The possibility of porting software written using PC software development tools is an excellent way to reduce development time in complex systems. This can be achieved by using a custom peripheral BIOS for the 186. A BIOS written to use the 80C186 peripherals would emulate the PC peripherals (timer, DMA, interrupt controller, serial communications) and could execute 'Well Behaved' PC code. 'Well Behaved' code is code that does not bypass the BIOS to access hardware directly. Some available PC software could run with little or no code modification. (Figure 3)

3) Hardware compatibility

Adding a standard chipset to the 80C186EC would allow 100% PC compatibility. This chipset could either be for an 8088 or 80286 processor. The chip set would provide all the PC peripherals and leave the 186 peripherals for control applications. Using an 8088 chipset would be very easy to implement but limited to a speed of 12MHz since the 8088 never reached speeds higher than that. To overcome the chipset speed limitation a 286 chipset could be used but decode logic would be needed to interface the 80C186 four state bus with the 286 two state bus. (Figure 4)

A possibility for the 80C186 or 80C186XL is the VADEM chip set (VG-501B & VG-502B) which also is PC compatible.

More details on these DOS interfaces will be provided in future issues.

Expanding 80C51SL-BG External Memory

Curt Durrant
Applications Engineer
Intel Corporation

Introduction

Most standard MCS®-51 products, when executing code from internal memory (ROM or EPROM) will automatically roll over into external code memory when the program counter reaches the end of the internal code space. The 80C51SL-BG, however, behaves somewhat differently. Although the differences are described in the 51SL data sheet, It would be helpful to elaborate somewhat for those customers planning to design with an internal ROM code that may also need to execute from external memory.

Internal vs External Code Memory

The 80C51SL-BG has an external address bus 15 bits wide and therefore can uniquely address 32K bytes of external memory. This is what will be referred to as the physical address space. The program counter, however is 16 bits wide and therefore has 64K bytes of unique addressable space. The program counter address will be referred to as the logical address.

When executing from internal memory (EAL = 1) the signal MEMCSL (used as a chip select for the external EPROM or code memory) is inactive for the first 32K of logical address space. Consequently, even though the logical address is the same as the physical address in this range, an external EPROM cannot be accessed unless the EPROM's chip select line is driven from some other source (such as grounding it). When the logical address reaches 8000h or above, MEMCSL goes low. Note that logical address 8000h is equivalent to physical address 0000h since internal A15 is not brought out to a package pin. When executing from external memory (EAL = 0), MEMCSL is always active.

External Data Memory / Flash Programming

Accessing external data memory is the same as a standard MCS-51 part for the first 32K of logical address space. Note however, that the 16 bytes located at address 7FF0 through 7FFF are not external data locations but rather are internal memory mapped registers for special functions. When accessing these registers the address will be present on the external bus, but the P36/WRL and P37/RDL signals will be disabled, preventing internal bus contention.

The data memory space from logical address 8000h to FFFFh is used for special FLASH programming functions. If external Flash memory is used for code storage this memory may be programmed and verified by doing data memory accesses at logical address 8000h and above. For example, when executing a Flash programming algorithm from internal memory, the instruction:

```
MOVX        @DPTR, A
```

where DPTR contains 8000h will cause the signal MEMCSL to go low (active), and the data memory write signal (P36/WRL) to go low. This will force a write to the Flash memory location 0000h with the data in the accumulator. To perform a verify, issue the instruction:

```
MOVX        A, @DPTR
```

where DPTR contains 8000h is used. The internal logic causes MEMCSL and PSENL to both go low, but prevents the data memory read signal (P37/RDL) from going low.

Note that when programming external Flash memory it is possible to do a write to both Flash and external data RAM (if present) simultaneously. If this is not acceptable then care must be taken to isolate the chip selects so that they

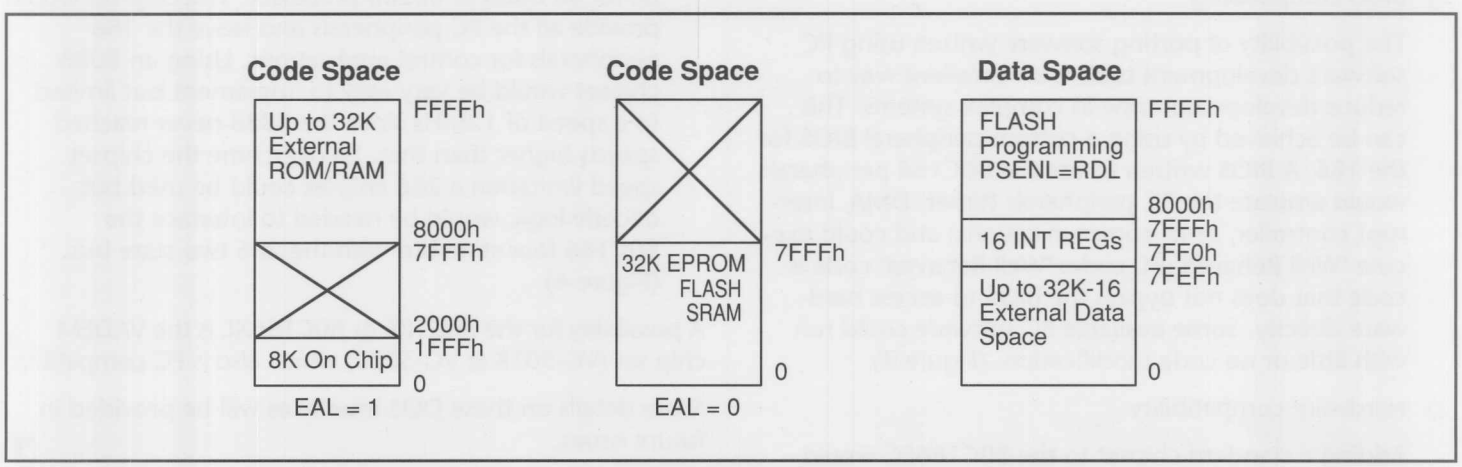


Figure 1. 80C51SL-BG Memory Map

will not both go active during Flash programming, but that they will both go active when appropriate during normal execution.

Conclusion

Even though external memory interfacing may seem somewhat unorthodox with the 80C51SL-BG once the user understands the signal conditions for various address ranges it becomes rather straight forward. The important issues to remember are:

1. With EAL = 1, MEMCSL is inactive for logical addresses 0 - 7FFFh, active for addresses 8000h to FFFFh. This is true for both code and data memory accesses.
2. With EAL = 0, MEMCSL is always active.
3. For data memory reads from logical address 8000h and above, P37/RDL is inactive and PSENL is active.

MCS®-96 Instruction Set Reference

This MCS-96 instruction set reference is included as an aid to writing or debugging your code. Note that instructions are not available on all devices. The following instructions are invalid for the 8X9X devices: BMOV, BMOVI, CMPL, DJNZW, DPTS, EPTS, IDLPD, POPA, PUSHA, TIJMP, XCH, and XCHB. Invalid instructions for the 8XC196KB devices are: BMOVI, DPTS, EPTS, TIJMP, XCH, and XCHB.

Hex Addr.	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xA	xB	xC	xD	xE	xF
0x	SKIP	CLR	NOT	NEG	XCH di	DEC	EXT	INC	SHR	SHL	SHRA	XCH ix	SHRL	SHLL	SHRAL	NORML
1x		CLRB	NOTB	NEGB	XCHB di	DECB	EXTB	INCB	SHRB	SHLB	SHRAB	XCHB ix				
2x	SJMP								SCALL							
3x	JBC								JBS							
	bit 0	bit 1	bit 2	bit 3	bit 4	bit 5	bit 6	bit 7	bit 0	bit 1	bit 2	bit 3	bit 4	bit 5	bit 6	bit 7
4x	AND 3op				ADD 3op				SUB 3op				MULU 3op *			
	di	im	in	ix	di	im	in	ix	di	im	in	ix	di	im	in	ix
5x	ANDB 3op				ADDB 3op				SUBB 3op				MULUB 3op *			
	di	im	in	ix	di	im	in	ix	di	im	in	ix	di	im	in	ix
6x	AND 2op				ADD 2op				SUB 2op				MULU 2op *			
	di	im	in	ix	di	im	in	ix	di	im	in	ix	di	im	in	ix
7x	ANDB 2op				ADDB 2op				SUBB 2op				MULUB 2op *			
	di	im	in	ix	di	im	in	ix	di	im	in	ix	di	im	in	ix
8x	OR				XOR				CMP				DIVU *			
	di	im	in	ix	di	im	in	ix	di	im	in	ix	di	im	in	ix
9x	ORB				XORB				CMPB				DIVUB *			
	di	im	in	ix	di	im	in	ix	di	im	in	ix	di	im	in	ix
Ax	LD				ADDC				SUBC				LDBZE			
	di	im	in	ix	di	im	in	ix	di	im	in	ix	di	im	in	ix
Bx	LDB				ADDCB				SUBCB				LDBSE			
	di	im	in	ix	di	im	in	ix	di	im	in	ix	di	im	in	ix
Cx	ST	BMOV	ST		STB	CMPL	STB		PUSH				POP	BMOVI	POP	
	di		in	ix	di		in	ix	di	im	in	ix	di		in	ix
Dx	JNST	JNH	JGT	JNC	JNVT	JNV	JGE	JNE	JST	JH	JLE	JC	JVT	JV	JLT	JE
Ex	DJNZ	DJNZW	TIJMP	BR in				LJMP					DPTS	EPTS	Reser-ved	LCALL
Fx	RET		PUSHF	POPF	PUSHA	POPA	IDLPD	TRAP	CLRC	SETC	DI	EI	CLRVT	NOP	signed multiply /divide*	RST

Note: not all instructions are implemented on all products. Consult User's Manuals.

* Signed multiplication and division are two-byte instructions. The first byte is "FE," and the second is the opcode of the corresponding unsigned instruction. For example, the opcode for MULU (3 operands) direct is "4C," so the opcode for MUL (3 operands) direct is "FE 4C."

im= immediate, di = direct, in = indirect, ix = indexed

Errata - Change Identifiers

How can a user tell what errata has been changed on an Embedded Control part? Change identifiers are the key. Change identifiers have been used on embedded products since 1990. The change identifier is the last character in the FPO number. The FPO number is typically a nine character number located on the second line of the top side package mark. Below is a list of errata, change identifiers and design considerations for embedded control products. Older devices do not have change identifiers. Devices that do not have change identifiers will have N/A in the change identifier field on this list.

MCS®-96 Family Errata

8X9XBH

The following errata covers just the last two most recent revisions of the 8X9XBH. Revisions are referred to as steppings. The last two steppings are the E-step and D-step. They are identified by an "E" or "D" respectively in the last character (change indicator) of the FPO number. The FPO number is the second line on the top side of the device. An example FPO number is: L1234567D. The "D" would indicate D-step. For a complete list of errata for all steppings call FaxBACK.

D-Step, Change Indicator (D)

Errata:

1) Indexed 3-operand multiply
2) HSI FIFO
3) Reserved location 2019H
4) RESET and the QBD pins
5) Software RESET timing
6) Using T2CLK for TIMER 2

E-Step, Change Indicator (E)

All D-step Errata not listed here was fixed in the E-Step device.

Errata:

1) Indexed 3-operand multiply
2) HSI Resolution
3) Reserved location 2019H
4) Reserved location 201CH

8XC196KB

The following errata covers just the last two most recent revisions of the 8XC196KB. Revisions are referred to as steppings. The last two steppings are the B and C step. They are identified by an "D" and "E" respectively in the last character (change indicator) of the FPO number. The FPO number is the second line on the top side of the device. An example FPO number is: L1234567D. The "D" would indicate B-step. For a complete list of errata for all steppings call FaxBACK.

B-Step, Change Indicator (D)

Errata:

1) Divide during HOLD or READY
2) HSI 8/9 state

C-Step, Change Indicator (E)

Errata:

1) HSI 8/9 State

8XC196KC

The following is a list of 8XC196KC errata and design considerations for all steppings of the device. Each entry is preceeded by a reference number. A complete explanation of these errata is available on FaxBACK, #2136.

A-step, Change Identifier (A) or No Change Identifier

Eratta:

101 A/D Convert Error
105 BMOVI Instruction
109 Divide Error During Hold
115 HSO IOC1 Bits Interchanged
126 Port0 Latched on Wrong Phase
131 PTS Req During INT Latency
132 NMI During PTS Latency
140 SIO Mode 0
143 TIJMP INDEX_MASK Value
150 Serial Port Framing Error
163 QBD Glitch During Powerup
173 A/D Linearity Too Large
174 Analog Input Latch-up
175 INST Weak During CCB Fetch
176 2 CCB Fetches
177 3 Waitstates During CCB
178 Pullups Too Weak During Reset
179 Buswidth Always 16-Bits
180 VCC Glitch Resets Device
181 Incomplete Reset At >12MHz
215 Oscillator Startup
216 Reset Hysteresis

Design considerations:
116 Indirect Shift Count Value
147 Write Cycle During Reset

B-1 step, Change Identifier (B)

All A-Step errata not listed here was fixed in the (B-1) Step device.

Eratta:
109 Divide Error During Hold
123 NMI during PTS Skips Address
163 QBD Glitch During Powerup
214 ONCE Mode Entry
215 Oscillator Startup
216 Reset Hysteresis

Design Considerations:
116 Indirect Shift Count Value
147 Write Cycle During Reset

B-3 step, Change Identifier (D) or (E)

All (B-1) Step eratta not listed here was fixed in the (B-3) Stepdevice.

Eratta:
109 Divide Error During Hold
123 NMI during PTS Skips Address
163 QBD Glitch During Powerup
216 Reset Hysteresis

Design Considerations:
116 Indirect Shift Count Value
147 Write Cycle During Reset

8XC196KR

Change Identifiers have been used on embedded products for the last two years. Change Identifiers are the last character in the FPO number. The FPO number is typically the 9 character number on the second line of the marking on top of the package.

A-Step, Change Identifier (A)

Errata:
1) Slave Programming Mode
2) EPA_MASK1/EPA_PEND1 written as words
3) BMOVI count register must reside in RAM < 128
4) TIJMP address holding register must be < 128
5) PTS interrupt during normal interrupt latency
6) Serial Port framing error bit does not work when MSB = 1
7) Remap function of EPA2 and EPA3 does not function correctly
8) A/D restart while conversion is active
9) PTS and NMI cannot be simultaneously active
10) Px_REG cleared when Px_MODE written on P2.7 & P6.4 - P6.7
11) Divide Error during HOLD/READY
12) SIO Mode 0 with BAUD contents > 8006H
13) EPAIPV value is multiplied by 2
14) Internal RAM during Powerdown
15) Oscillator sensitivity

Design Considerations:
1) EPA timer counters must decrement/increment to match value
2) P6.4 - P6.7 must not be modified while P6_MODE.x = 1
3) CLKOUT does not function as open-drain
4) EPA Timer write should not conflict with Timer reset
5) Device reset should not occur during an external memory write
6) Indirect shifts should not be greater than 32

B-Step, Change Identifier (C)

Errata:
1) I_{OH2} = - 6µA

Design Considerations:
1) EPA timer counters must decrement/increment to match value
2) P6.4 - P6.7 must not be modified while P6_MODE.x = 1
3) CLKOUT does not function as open-drain
4) EPA Timer write should not conflict with Timer reset
5) Device reset should not occur during an external memory write
6) Indirect shifts should not be greater than 32

80C186 Family Errata

80C186

The 80C186 is currently on the B-stepping. The A and B step products have identical errata.

Errata:

1) Non-contiguous Interrupt Acknowledge cycles
2) ERROR# processing during FWAIT instructions
3) Input high voltage requirement on SRDY and ARDY pins
4) Interrupt Status Register (DHLT and Timer Interrupts)
5) Bus Preemption Bug (HOLD/HOLDA protocol)
6) 80C188 RFSH# pin output timing

80C186EA / 80L186EA

A-Step, Change Identifier (A) or ( )

1) Low hysteresis on RESIN# pin
2) TEST/BUSY#, RD#/QSMD#, LCS#, and UCS# input low voltage
3) INT1/INTA1 in Cascade Mode

B-Step, Change Identifier (B)

1) INT1/INTA1 in Cascade Mode

80C186EB / 80L186EB

A-Step, Change Identifier (A) or ( )

1) Entry into ONCE Mode
2) Low hysteresis on RESIN# pin
3) SINT1 input not latched internally
4) Ready input during INTA# bus cycles
5) CLKOUT transitions on the rising edge on CLKIN instead of the falling edge
6) I/O Ports 1 and 2 initialize to Port instead of Peripheral function (documentation error)
7) INT1/INTA1 in Cascade Mode

B-Step, Change Identifier (B) or (C)

1) INT1/INTA1 in Cascade Mode

80C186EC

A-Step, Change Identifier (A) or ( )

1) Early exit from Reset (w/ high Vcc and/or low temperature)
2) Powersave Mode initialization at Reset

B-Step, Change Identifier (b)

1) No Known Errata

80C186XL

A-Step, Change Identifier (A) or ( )

Errata:

1) Low Hysteresis on RESIN# pin
2) TEST/BUSY#, RD#/QSMD#, LCS#, and UCS# input low voltage
3) INT1/INTA1 in Cascade Mode

B-Step, Change Identifier (B)

Errata:

1) INT1/INTA1 in Cascade Mode

Sales Offices and Distributor Addresses

Intel Corporation
Literature Department
2200 Mission College Blvd.
P.O. Box 58119
Santa Clara, CA 95052-8119
United States

Intel Japan K.K.
5-6 Tokodai, Tsukuba-shi
Ibaraki, 300-26
Japan

Intel Corporation S.A.R.L.
1, Rue Edison, BP 303
78054 Saint-Quentin-en-
Yvelines Cedex
France

Intel Corporation (U.K.) Ltd.
Pipers Way
Swindon
Wiltshire, England SN3 1RJ
United Kingdom

Intel GmbH
Dornacher Strasse 1
8016 Feldkirchen bei Muenchen
Germany

Intel Semiconductor Ltd.
10/F East Tower
Bond Center
Queensway, Central
Hong Kong

Intel Semiconductor of
Canada, Ltd.
190 Attwell Drive, Suite 500
Rexdale, Ontario M9W 6H8
Canada

U.S. INTEL SALES OFFICES:

Alabama, Huntsville, (205) 883-3507
Arizona, Phoenix, (602) 231-0386
California, Roseville, (916) 782-8086; San Diego, (619) 292-8086; Santa Ana, (714) 835-9642; Santa Clara, (408) 986-8086; Sherman Oaks, (310) 640-6040
Colorado, Denver, (303) 321-8086
Connecticut, Danbury, (203) 748-3130
Florida, Deerfield Beach, (305) 421-0506; Orlando, (407) 851-1987
Georgia, Norcross, (404) 449-0541
Illinois, Schaumburg, (708) 605-8031
Indiana, Indianapolis, (317) 875-0623
Maryland, Annapolis Junction, (410) 206-2860
Massachusetts, Westford, (508) 692-0960
Michigan, West Bloomfield, (313) 851-8096
Minnesota, Bloomington, (612) 835-6722
New Jersey, Red Bank, (908) 747-2233
New York, Fairport, (716) 425-2750; Fishkill, (914) 897-3860; Islandia, (516) 231-3300
Ohio, Dayton, (513) 890-5350; Beachwood, (216) 292-8224
Oklahoma, Oklahoma City, (405) 848-8086
Oregon, Beaverton, (503) 645-8051
Pennsylvania, Blue Bell, (215) 641-1000; Pittsburgh, (412) 823-4970
Puerto Rico, Las Piedras, (809) 733-8616
South Carolina, Greenville, (803) 297-8086
Texas, Austin, (512) 794-8086; Dallas, (214) 241-8087; Houston, (713) 988-8086
Utah, Murray, (801) 263-8051
Washington, Bellevue, (206) 643-8086, Spokane, (509) 928-8086
Wisconsin, Brookfield, (414) 789-2733

CANADIAN SALES OFFICES:

British Columbia, Intel Semiconductor of Canada, Ltd., Vancouver, (604) 844-2823
Ontario, Intel Semiconductor of Canada, Ltd., Ottawa, (613) 829-9714; Intel Semiconductor of Canada, Ltd., Rexdale, (416) 675-2105
Quebec, Intel Semiconductor of Canada, Ltd., Pt. Claire, (514) 694-9130

U.S. DISTRIBUTORS:

Alliance Electronics, Inc. / Almac/Arrow Electronics / Arrow Commercial Systems Group / Arrow/Schweber Electronics / Avnet Computer / Hamilton/Avnet / MTI Systems / MTI Systems Sales / North Atlantic Industries Systems Division / Pioneer-Standard / Pioneer Technologies Group / Wyle Laboratories

CANADIAN DISTRIBUTORS:

Almac-Arrow Electronics / Arrow/Schweber Electronics / Avnet Computer / Hamilton/Avnet / Zentronics

EUROPEAN INTEL SALES OFFICES:

Finland, Helsinki, (358) 0 544 644
France, St. Quentin-en-Yvelines Cedex, (33) (1) 30 57 70 00
Germany, Feldkirchen bei Muenchen, (49) 089/90992-0
Israel, Tel-Aviv, (972) 03 498080
Italy, Assago, (39) (02) 89200950
Netherlands, Rotterdam, (31) 10 407 11 11
Spain, Madrid, (34) 308 25 52
Sweden, Solna, (46) 8 734 01 00
United Kingdom, Swindon, (44) (0793) 696000

EUROPEAN DISTRIBUTORS:

Austria, Bacher Electronics GmbH, Wien, 43 222 81356460
Belgium, Inelco Belgium S.A., Bruxelles, 32 2 244 2811
France, Almex, Antony Cedex, 33 1 4096 5400; Lex Electronics, Rungis Cedex, 33 1 4978 4978; Tekelec, Sevres, 33 1 4623 2425
Germany, E2000 Vertriebs-AG, Muenchen, 49 89 420010; Jermyn GmbH, Limburg, 49 6431 5080; Proelectron Vertriebs GmbH, Dreieich, 49 6103 304343
Greece, Pouliadis Associates Corp., Athens, 30 1 360 3741
Ireland, Micro Marketing, Dublin, 010 3531 989 400
Israel, Eastronics Ltd., Tel-Aviv, 972 3 6458777

EUROPEAN DISTRIBUTORS (Contd.):

Italy, Intesi Div. Della Deutsche, Assago (Milano), 39 2 824701; Lasi Elettronica S.P.A., Milano, 39 2 66101370
Netherlands, Datelcom, AP Delft, 31 15 609 906; Diode Components b.v., NG Nieuwegein, 3402 91234
Scandinavia, OY Fintronic AB, Finland, 358 0 6926022; ITT Multikomponent A/S, Denmark, 45 42 451822; Nortec Electronics A/S, Norway, 47 2 846210; Nortec Electronics AB, Sweden, 46 8 7051850
South Africa, EBE, Silverton, 27 12 803 7680
Spain, ATD Electronica, S.A., Madrid, 34 1 661 6551
Switzerland, Industrade A.G., Wallisellen, 41 1 8328111
Turkey, EMPA, Istanbul, 901 5993050
United Kingdom, Avnet-Access, Hertsfordshire, 0462 480888; Bytech Components Ltd., Hants, 0256 707107; Jermyn, Kent, 0732 743743; MMD/Rapid, Berkshire, 0734 313232
Yugoslavia, H.R. Microelectronics Corp., Santa Clara, CA, U.S.A., (408) 988-0286

INTERNATIONAL SALES OFFICES:

Australia, Intel Australia Pty. Ltd., Sydney, 61-2-975-3300; Intel Australia Pty. Ltd., Melbourne, 61-3-810-2141
Brazil, Intel Semiconductores do Brazil LTDA, Sao Paulo, 55-11-287-5899
China/Hong Kong, Intel PRC Corp., Beijing, (1) 500-4850; Intel Semiconductor Ltd., Hong Kong, (852) 844-4555
India, Intel Asia Electronics, Inc., Bangalore, 91-812-215773
Japan, Intel Japan K.K., Tsukuba HQ, 0298-47-8511; Intel Japan K.K., Tokyo HQ, 03-3201-3621; Intel Japan K.K., Saitama, 0485-24-6871; Intel Japan K.K., Kanagawa, 045-474-7660; Intel Japan K.K., Osaka, 06-863-1091; Intel Japan K.K., Hachioji-shi, 0426-48-8770; Intel Japan K.K., Aichi, 052-204-1261
Korea, Intel Korea, Ltd., Seoul, (2) 784-8186
Singapore, Intel Singapore Technology, Ltd., (65) 250-7811
Taiwan, Intel Technology Far East Ltd., Taipei, 886-2-5144202

INTERNATIONAL DISTRIBUTORS:

Argentina, Dafsys S.R.L., Buenos Aires, 54.1334.1871
Australia, Email Electronics, Huntingdale, 011-61-3-544-8244; NSD-Australia, Victoria, 03 8900970
Brazil, Microlinear, Sao Paulo, 5511-220-2215
Chile, Sisteco, Santiago, 562-234-1644
China/Hong Kong, Novel Precision Machinery Co., Ltd., Kowloon, Hong Kong, (852) 360-8999
Guatemala, Abinitio, Guatemala City, 5022-32-4104
India, Priya International Limited, Bangalore, (91) 812-214027, 812-214395; Priya International Limited, Bombay, (91) 22-2863611, 22-2863676, 22-2863900, 22-2864026; Priya International Limited, New Delhi, (91) 11-3314512, 11-3310413; Priya International Limited, Madras, (91) 44-451031, 44-451597; Priya International Limited, Secunderabad, (91) 842-70220, 842-77059; SES Computers and Technologies Pvt. Ltd., Bangalore; S&S Corp., San Jose, CA, U.S.A., (408) 978-6216
Jamaica, MC Systems, Kingston, (809) 929-2638, (809) 926-0188
Japan, Asahi Electronics Co. Ltd., Kitakyushu-shi, 093-511-6471; CTC Components Systems Co., Ltd., Kanagawa 213, 044-852-5121; Dia Semicon Systems, Inc., Tokyo 154, 03-3439-1600; Okaya Koki, Nagoya-shi, 052-204-8315; Ryoyo Electro Corp., Tokyo 104, 03-3546-5011
Korea, J-Tek Corp., Seoul, (822) 557-8039; Samsung Electronics, Seoul, (822) 751-3680
Mexico, PSI S.A. de C.V., Cuernavaca-MOR, 52-73-13-9412, 52-73-17-5340
New Zealand, Email Electronics, Penrose, Auckland, 011-64-9-591-155
Saudi Arabia, AAE Systems, Inc., Sunnyvale, CA U.S.A., (408) 732-1710
Singapore, Electronic Resources Pte, Ltd., (65) 283-0888
South Africa, Electronic Building Elements, Pretoria, 011-2712-803-7680
Taiwan, Micro Electronics Corp., Taipei, R.O.C., (886) 2-7198419; Acer Sertek, Inc, Taipei, R.O.C., (886) 2-501-0055
Uruguay, Interfase, Montevideo, 5982-96-0490, 5982-96-1143
Venezuela, Unixel C.A., Caracas, 582-238-6082

CG/SODA/070892